MINESnancy

Analyse Numérique

Cours de Antoine Henrot

Cours Tronc Commun Scientifique TCSS5AA

Semestre S5 : 2016-2017

Ecole des Mines de Nancy - Campus Artem - 54 042 Nancy Cedex
Tel : 03 55 66 28 10 - email: antoine.henrot@univ-lorraine.fr

Introduction

Voici le polycopié du cours de Mathématiques I qui porte sur l'analyse numérique. Il a pour origine un cours d'analyse numérique rédigé par Michel Pierre il y a quelques années, mis en forme sous LaTeXpar Manon Didry, puis enrichi et modifié en continu depuis.

Chaque chapitre correspond à une séance de cours. Dans le polycopié, chaque chapitre est structuré de la façon suivante :
- D'abord le cours proprement dit, que vous devez lire avant la séance.
- Puis une liste d'exercices plutôt de nature théorique, que vous pouvez faire pour vous entraîner.
- L'énoncé du TD correspondant au chapitre, TD que vous réaliserez en salle machine l'après-midi.
- Enfin, une petite fiche "aide-mémoire" qui pourra être utile à certains pour préciser des points relatifs aux mathématiques sous-jacentes au chapitre.

Pour résumer : pour lundi 19 septembre, date du premier cours, il vous est demandé de lire le premier chapitre et de noter sur une feuille les questions que vous souhaitez poser durant la séance. En effet la première heure en Amphi est consacrée pour partie à la réponse aux questions des élèves. Durant les deux heures suivantes, vous ferez, en salle machine, des exercices et des petits programmes en Matlab pour illustrer les notions vues en cours.

Antoine Henrot

Table des matières

1 Les erreurs en Analyse Numérique. Notions de conditionnement et stabilité **11**
1.1 Introduction . . . 11
1.2 Arithmétique machine . . . 12
1.2.1 Représentation machine des nombres réels . . . 12
1.2.2 Perte de chiffres significatifs . . . 12
1.3 Notions de conditionnement et stabilité . . . 14
1.3.1 Conditionnement . . . 15
1.3.2 Stabilité . . . 16
1.3.2.1 Algorithme implicite . . . 17
1.4 Conclusion . . . 18
1.5 Exercices du chapitre 1 . . . 19
1.6 TD1 : Initiation Matlab . . . 20
1.7 Aide-mémoire . . . 22
1.7.1 Formules de Taylor . . . 22
1.7.2 Intégration par parties . . . 22

2 Interpolation et approximation polynômiale **23**
2.1 Interpolation de Lagrange . . . 23
2.1.1 Le polynôme d'interpolation . . . 23
2.1.2 Propriétés des différences divisées . . . 25
2.1.3 Erreur dans l'interpolation de Lagrange . . . 27
2.1.3.1 Analyse du Théorème 2.3 . . . 28
2.2 Approximation au sens des moindres carrés . . . 30
2.2.1 Introduction . . . 30
2.2.2 Méthode classique des moindres carrés . . . 31
2.2.3 Polynômes orthogonaux . . . 34
2.2.4 Exemples classiques de polynômes orthogonaux . . . 36
2.2.5 Polynômes trigonométriques . . . 38
2.2.6 La transformée de Fourier rapide . . . 41
2.3 Approximation par des fonctions polynômiales par morceaux . . . 42
2.3.1 Approximation linéaire par morceaux . . . 42

2.3.2 Approximation cubique . . . 42
2.3.3 Fonctions splines . . . 43
2.4 Exercices du chapitre 2 . . . 45
2.5 TD2 : Interpolation et moindres carrés . . . 47
2.6 Aide-mémoire . . . 49
2.6.1 Polynômes . . . 49
2.6.2 Théorème des accroissements finis . . . 49
2.6.3 Norme . . . 49

3 Dérivation et intégration numérique 51
3.1 Introduction . . . 51
3.2 Dérivation numérique . . . 52
3.2.1 Dérivée première . . . 52
3.2.1.1 Formules à deux points . . . 53
3.2.1.2 Formules à trois points . . . 53
3.2.1.3 Erreur . . . 54
3.2.2 Dérivées d'ordre supérieur . . . 54
3.2.3 Méthode des différences finies . . . 54
3.2.3.1 Principe de la méthode . . . 54
3.2.3.2 Autre exemple : l'équation de la chaleur . . . 56
3.2.3.3 Stabilité . . . 57
3.3 Intégration numérique : méthodes composites . . . 59
3.3.1 Principe . . . 59
3.3.2 Méthode des rectangles . . . 59
3.3.3 Méthode des trapèzes . . . 60
3.3.4 Méthode de Simpson . . . 61
3.4 Analyse de l'erreur dans les méthodes d'intégration . . . 62
3.4.1 Théorie . . . 62
3.4.2 Application aux méthodes usuelles . . . 63
3.4.2.1 Méthode des rectangles (à gauche ou à droite) . . . 63
3.4.2.2 Formule du point milieu . . . 63
3.4.2.3 Méthode des trapèzes . . . 64
3.4.2.4 Méthode de Simpson . . . 64
3.4.3 Exemple d'application . . . 65
3.5 Méthodes d'intégration numérique de Gauss . . . 65
3.5.1 Introduction . . . 65
3.5.2 Idée de base des méthodes de Gauss . . . 66
3.5.3 Mise en œuvre de la méthode . . . 67
3.5.3.1 Méthode de Legendre-Gauss . . . 68
3.5.3.2 Méthode de Gauss-Chebyshev . . . 69
3.5.3.3 Méthodes de Laguerre et Hermite . . . 69

3.6 Exercices du chapitre 3 . . . 70
3.7 TD3 : Intégration numérique et méthode des différences finies . . . 71
3.8 Aide-mémoire . . . 72
3.8.1 Utilisation d'une primitive . . . 72
3.8.2 Intégration par parties . . . 72
3.8.3 Changement de variables . . . 72

4 Équations différentielles - Théorie et méthodes d'Euler 73
4.1 Quelques résultats théoriques sur l'équation différentielle (4.1) . . . 73
4.1.1 Quelques soucis d'ordre théorique . . . 74
4.1.2 Régularité de la solution . . . 77
4.2 Méthodes de résolution à un pas . . . 77
4.2.1 La méthode d'Euler . . . 77
4.2.1.1 Estimation de l'erreur de discrétisation . . . 79
4.2.1.2 Influence des erreurs d'arrondis . . . 81
4.2.2 Méthode d'Euler implicite . . . 83
4.3 Résolution d'EDO en utilisant Matlab . . . 84
4.4 Exercices du chapitre 4 . . . 84
4.5 TD4 : Résolution numérique d'équations différentielles . . . 86
4.6 Aide-mémoire : équations différentielles linéaires . . . 87

5 Équations différentielles - Méthodes de Runge-Kutta et méthodes multi-pas 89
5.1 Les méthodes de Runge-Kutta . . . 89
5.2 Contrôle du pas . . . 93
5.3 Méthodes à pas multiples . . . 95
5.3.1 Méthodes d'Adams-Bashforth à $r+1$ pas . . . 96
5.3.2 Méthodes d'Adams-Moulton à $r+1$ pas . . . 97
5.3.3 Méthode de prédicteur-correcteur . . . 98
5.4 Comparaison des méthodes . . . 99
5.5 Applications à des problèmes aux limites . . . 100
5.6 Exercices du chapitre 5 . . . 101

6 Résolution de systèmes linéaires - méthodes directes 103
6.1 Quelques remarques générales . . . 103
6.1.1 Origines des problèmes . . . 103
6.1.2 Méthodes utilisées . . . 104
6.1.3 Structure de la matrice . . . 104
6.1.3.1 Cas d'une matrice diagonale . . . 104
6.1.3.2 Matrice triangulaire supérieure (ou inférieure) . . . 104
6.1.3.3 Autres cas . . . 105
6.1.3.4 Matrices symétriques . . . 106

6.1.4 Stockage des matrices . . . 107
6.2 Méthodes directes de résolution de $AX = b$ . . . 110
6.2.1 Méthode d'élimination de Gauss . . . 110
6.2.1.1 Calcul du nombre d'opérations élémentaires . . . 113
6.2.2 Dérivés de la méthode de Gauss . . . 113
6.2.2.1 Factorisation LU . . . 113
6.2.2.2 Algorithme de Crout . . . 115
6.2.2.3 Cas particulier des matrices tridiagonales . . . 117
6.2.2.4 Méthode de Gauss-Jordan . . . 118
6.2.3 Factorisation de Cholesky . . . 119
6.2.4 Autres méthodes directes . . . 121
6.3 Exercices du chapitre 6 . . . 121
6.4 TD6 : résolution des systèmes linéaires . . . 123
6.5 Aide-mémoire : Rappels sur les matrices . . . 125

7 Résolution de systèmes linéaires - méthodes itératives 127
7.1 Analyse du conditionnement d'un système linéaire . . . 127
7.1.1 Quelques préliminaires sur les normes matricielles . . . 127
7.1.1.1 Norme sur l'espace vectoriel des matrices . . . 128
7.1.2 Analyse du conditionnement . . . 130
7.1.2.1 Propriétés de $cond\,(A)$ . . . 131
7.1.3 Exemple de système linéaire mal conditionné . . . 132
7.1.4 Préconditionnement . . . 133
7.2 Méthodes itératives . . . 133
7.2.1 Description des méthodes de Jacobi, Gauss-Seidel, relaxation . . . 134
7.2.2 Itérations par blocs . . . 135
7.2.3 Etude de la convergence des méthodes itératives . . . 136
7.2.3.1 Vitesse de convergence . . . 137
7.2.3.2 Application aux méthodes de Jacobi, Gauss-Seidel et Relaxation . . . 138
7.2.4 Application à la méthode des différences finies . . . 142
7.3 Autres méthodes itératives modernes . . . 146
7.4 Exercices du chapitre 7 . . . 147
7.5 TD7 : méthodes itératives . . . 148
7.6 Aide-mémoire : Rappels sur les valeurs propres . . . 149

8 Résolution d'équations et systèmes non linéaires 151
8.1 Introduction . . . 151
8.2 Cas des fonctions d'une variable . . . 151
8.2.1 Préliminaires : séparation des zéros . . . 151
8.2.2 Quelques algorithmes classiques . . . 152

8.2.2.1 Méthode de dichotomie (ou bisection) . . . 152
8.2.2.2 Méthode de la sécante . . . 153
8.2.2.3 Critère d'arrêt . . . 154
8.2.2.4 Méthode de Newton . . . 154
8.2.2.5 Méthode de point fixe . . . 155
8.2.3 Convergence des algorithmes . . . 156
8.2.3.1 Méthodes de point fixe . . . 156
8.2.3.2 Méthode de Newton . . . 159
8.2.3.3 Méthode de la sécante . . . 161
8.2.4 Comparaison des algorithmes . . . 163
8.2.4.1 Méthode de dichotomie . . . 163
8.2.4.2 Méthode de Newton . . . 164
8.2.4.3 Méthode de la sécante . . . 164
8.2.4.4 Méthode du point fixe . . . 164
8.2.5 Accélération de la convergence . . . 166
8.2.5.1 Méthode de relaxation . . . 167
8.2.5.2 Méthode du Δ^2 d'Aitken . . . 167
8.3 Systèmes d'équations non linéaires . . . 170
8.3.1 La méthode de Newton-Raphson . . . 171
8.3.2 Méthodes de type quasi-Newton . . . 172
8.4 Exercices du chapitre 8 . . . 172
8.5 TD8 : recherche de zéros de fonctions à une ou plusieurs variables . . . 174

9 Eléments sur l'optimisation 175
9.1 Aspects théoriques . . . 175
9.1.1 Introduction . . . 175
9.1.2 Résultats d'existence . . . 175
9.1.3 Convexité . . . 176
9.1.4 Conditions d'optimalité . . . 177
9.1.4.1 Contraintes égalités et inégalités . . . 179
9.2 Algorithmes d'optimisation mono-dimensionnels ou recherche du pas . . . 180
9.2.1 Méthode de la section dorée . . . 180
9.2.2 Méthode d'interpolation parabolique . . . 181
9.2.3 D'autres règles . . . 182
9.3 Quelques notions sur les algorithmes . . . 183
9.4 Méthodes de gradient . . . 184
9.5 Méthode du gradient conjugué . . . 185
9.6 Approche par la méthode de Newton . . . 189
9.7 Exercices du chapitre 9 . . . 189
9.8 TD9 : optimisation . . . 190
9.9 Aide-mémoire . . . 192

9.9.1 Topologie 192
9.9.2 Calcul différentiel 193
9.9.2.1 Différentiabilité 193
9.9.2.2 Dérivées partielles 193
9.9.3 Convexité 194

Chapitre 1

Les erreurs en Analyse Numérique. Notions de conditionnement et stabilité

1.1 Introduction

Dans le titre de ce chapitre, le mot **erreur** n'est pas pris au sens de faute (raisonnement faux dans la méthode, instruction fausse dans le programme). Il ne concerne que des erreurs *inévitables*. On peut les classer en trois catégories :

- **Les erreurs sur les données**. Elles peuvent être dues à l'imprécision des mesures physiques ou au fait que les données proviennent elle même d'un calcul approché. Elles sont imposées, en quelque sorte, de l'extérieur et nous ne pouvons agir sur elles. Néanmoins, la manière dont elles se propagent au cours des calculs est davantage du ressort du calculateur. L'analyse de cette propagation sera évoquée au cours de ce chapitre. Elle est liée aux notions de conditionnement et de stabilité (voir section 1.3).
- **Les erreurs d'arrondi**. Ce sont les erreurs dues au fait que la machine (dans ce cours, ce terme désignera indifféremment la calculette de poche ou l'ordinateur) ne peut représenter les nombres réels qu'avec un nombre fini de chiffres. A chaque opération mathématique élémentaire, il pourra y avoir une perte de chiffres significatifs. Le calculateur doit donc être vigilant quand le nombre d'opérations est très important. Cela va faire l'objet du prochain paragraphe.
- **Les erreurs d'approximation ou de discrétisation**. Ce sont les erreurs qu'on commet, par exemple, lorsqu'on calcule une intégrale à l'aide d'une somme finie, une dérivée à l'aide de différences finies ou bien la somme d'une série infinie à l'aide d'un nombre fini de ses termes (on parle alors quelquefois d'erreur de troncature). Une situation qu'on rencontrera souvent également consiste à approcher une fonction, solution d'une certaine équation fonctionnelle ou aux dérivées partielles, par une combinaison linéaire finie de fonctions élémentaires. Ce type d'erreurs est bien sûr fortement lié à la méthode employée. Un des buts de l'analyse numérique consiste justement à évaluer ces erreurs de discrétisation pour chaque algorithme mis en place. C'est donc un souci qui nous accompagnera tout au long de ce cours.

1.2 Arithmétique machine

1.2.1 Représentation machine des nombres réels

Le système de représentation machine des nombres réels le plus utilisé en calcul scientifique est celui de la représentation en virgule flottante normalisée. Un nombre $x \neq 0$ s'écrit

$$x \approx \pm m\, b^p$$

où b est la base de numération (entier supérieur ou égal à 2), m la mantisse (ensemble de chiffres de la base b) et p l'exposant (entier relatif). Dire que la représentation est normalisée, c'est supposer

$$b^{-1} \leq m < 1.$$

Par exemple, le nombre 395.2134 en base 10 s'écrit $+.3952134\ 10^{+3}$ ou, plus souvent, $+.3952134\ E{+}3$. Les bases $b = 2$ ou les puissances de 2 sont fréquemment utilisées par les constructeurs d'ordinateurs. Cela signifie, par exemple que tous les calculs internes sont faits en base 2 et seul le résultat affiché est traduit en base 10.
La mantisse m est donc un nombre qui commence toujours par 0. ... et qui s'écrit avec un nombre maximum N de chiffres significatifs (imposé par le choix de la taille des emplacements mémoires alloués au type *réel*). Signalons la possibilité offertes sur la plupart des ordinateurs (mais pas sur les calculatrices !) de travailler en *double précision*, c'est-à-dire essentiellement avec une mantisse comportant $2N$ chiffres significatifs. Cette possibilité est évidemment coûteuse en temps de calcul et ne doit surtout pas être utilisée systématiquement. Elle est intéressante, en particulier, quand on n'est pas très sûr de la stabilité d'un algorithme, pour pouvoir comparer des résultats obtenus en simple précision et en double précision.
L'exposant p est lui aussi limité par la machine avec laquelle on travaille. Si p est hors des limites permises (i.e. trop grand ou trop petit en valeur absolue), le nombre x ne sera pas représentable en machine : elle affichera alors un message du type *exponent overflow* ou plus simplement Inf (Infini) quand le nombre est trop grand, ou elle affichera 0 s'il est trop petit, voir Exercice 1.1. On appelle quelquefois **capacité** le plus grand nombre avec lequel peut travailler la machine et **pas** le plus petit nombre. Par exemple en Matlab, *realmin* désigne le pas, *realmax* la capacité.

La différence entre la valeur exacte d'un nombre x et la valeur x^* de sa représentation est appelée **erreur d'arrondi**. Elle est majorée par $\frac{1}{2}b^{-N}b^p$, soit en valeur relative :

$$\left|\frac{x - x^*}{x}\right| \leq \frac{b^{-N}b^p}{2x} \leq \frac{b^{-N}b^p}{2b^{p-1}} = \frac{b^{1-N}}{2}\,.$$

Cette erreur est systématiquement présente dans tout calcul arithmétique sur nombre réel effectué par un ordinateur. Elle fait que l'arithmétique numérique n'a pas la précision de l'arithmétique mathématique et peut, si on n'y prend garde, conduire à des résultats inexacts, voire aberrants, comme le montrent les exemples suivants et les exercices.

1.2.2 Perte de chiffres significatifs

Pour faciliter la compréhension, nous nous placerons dans l'environnement rassurant de la base 10, les phénomènes étant du même type dans toute base.

Exemple 1.1 Supposons que dans un calcul apparaisse la quantité $x = \pi - 3.1415$ (où $\pi = 3.141592653589793\ldots$). Si on travaille avec 8 chiffres significatifs (comme beaucoup de calculettes), le nombre π sera représenté par : $\pi^* = 0.31415927\ 10^{-1}$ en virgule flottante normalisée. On aura donc

$$x = 0.31415927\ 10^{-1} - 0.31415\ 10^{-1} = 0.0000927\ 10^{-1} = 0.927\ 10^{-4}$$

On constate que x ne contient en fait que 3 chiffres significatifs et non 8, soit une perte sèche de 5 chiffres significatifs. Ce genre de phénomènes peut surgir à tout moment d'un calcul. Nous verrons de nombreux exemples d'algorithmes conduisant à faire la différence de 2 nombres proches.

Exemple 1.2 : Soit à calculer le quotient

$$A = \frac{XN}{XD} = \frac{\pi - 3,1415}{10^4(\pi - 3,1515) - 0,927}$$

En travaillant avec 8 chiffres significatifs, on a :

$$XD\ = 10^4(0,927.10^{-4}) - 0,927 = 0,0$$

$\pi^* = 3,1415927$	$A = ERREUR$
$\pi^* = 3,14159265$	$A = -0,18530...$
$\pi^* = 3,141592653\ \rceil$	$A = -0,197134...$
$\pi^* = 3,141592654\ \rfloor$	$A = -0,201427...$
$\pi^* = 3,1415926535\ \rceil$	$A = -0,1992548...$
$\pi^* = 3,1415926536\ \rfloor$	$A = -0,1996844...$
$\pi^* = 3,14159265358\ \rceil$	$A = -0,1995984...$
$\pi^* = 3,14159265359\ \rfloor$	$A = -0,19964143...$
$\pi^* = 3,141592653589$	$A = -0,19963713...$
$\pi^* = 3,1415926535897\ \rceil$	$A = -0,199640143...$
$\pi^* = 3,1415926535898\ \rfloor$	$A = -0,1996405743...$
$\pi^* = 3,14159265358979$	$A = -0,1996405312$
$\pi^* = 3,1415927653589793$	$A = -0,1996405439...$

On constate que pour obtenir p chiffres significatifs pour A, il faut représenter π à l'aide de $p + 8$ chiffres significatifs.
Dans le tableau ci dessus, le symbole $\rceil$ signale des représentations avec le même nombre de chiffres, tous étant identiques sauf le dernier.

Exemple 1.3 : L'addition numérique n'est pas associative : en virgule flottante à n chiffres significatifs, $(a + b) + c$ peut être différent de $a + (b + c)$. C'est le cas avec le choix suivant où les calculs sont faits avec 8 chiffres significatifs.

$$\begin{aligned} a &: = 0,23371258.10^{-4} \\ b &: = 0,33678429.10^{2} \\ c &: = -0,33677811.10^{2} \end{aligned}$$

$$a + b = 0,00000023(371258).10^2 + 0,33678429.10^2 = 0,33678452.10^2.$$

On remarque que, dans cette addition, les 6 derniers chiffres de a sont perdus. Ainsi :

$$\begin{aligned}(a+b)+c &= 0,33678452.10^2 - 0,33677811.10^2 \\ &= 0,00000641.10^2 = 0,641.10^{-3}.\end{aligned}$$

Par ailleurs :

$$\begin{aligned}b+c &= 0,33678429.10^2 - 0,33677811.10^2 \\ &= 0,00000618.10^2 = 0,618.10^{-3}\end{aligned}$$

$$a+(b+c) = 0,02337125\,(8)\,.10^{-3} + 0,61800000.10^{-3} = 0,64137126.10^{-3}.$$

Essayons d'analyser le phénomène ci-dessus. Si $vf\,(a+b)$ désigne le résultat de l'addition $a+b$, on a :

$$vf\,(a+b) = (a+b)\,(1+\varepsilon_1)$$

où ε_1 désigne l'erreur relative commise dans le calcul : celle-ci dépend de la précision de la machine. On sait qu'elle est majorée par $\frac{1}{2}\beta^{1-n}$ soit ici 5.10^{-8}. Posons $\eta = vf\,(a+b)$. On a alors

$$\begin{aligned}vf\,((a+b)+c) &= vf\,(\eta+c) = (\eta+c)\,(1+\varepsilon_2) \qquad |\varepsilon_2| \leq 5.10^{-8} \\ &= [(a+b)\,(1+\varepsilon_1)+c]\,(1+\varepsilon_2) \\ &= a+b+c+(a+b)\,\varepsilon_1\,(1+\varepsilon_2)+(a+b+c)\,\varepsilon_2.\end{aligned}$$

Ainsi :

$$\frac{vf\,((a+b)+c)-(a+b+c)}{a+b+c} = \frac{a+b}{a+b+c}\varepsilon_1\,(1+\varepsilon_2)+\varepsilon_2.$$

De la même façon :

$$\frac{vf\,(a+(b+c))-(a+b+c)}{a+b+c} = \frac{b+c}{a+b+c}\varepsilon_3\,(1+\varepsilon_4)+\varepsilon_4.$$

On voit que les erreurs $\varepsilon_1\,(1+\varepsilon_2)$, $\varepsilon_3\,(1+\varepsilon_4)$, environ égales à 5.10^{-8} sont soumises à des coefficients amplificateurs

$$\frac{a+b}{a+b+c} \simeq 5.10^4, \qquad \frac{b+c}{a+b+c} \simeq 0,9.$$

Ceci explique pourquoi le second calcul est plus précis que le premier.

Remarque 1.1 Dans les calculs où interviennent des nombres d'ordres de grandeur différents, il est en général préférable d'effectuer les opérations en groupant ceux d'ordres de grandeur similaires pour éviter les pertes de chiffres significatifs.

1.3 Notions de conditionnement et stabilité

Ces deux notions, toujours présentes en analyse numérique, sont relatives à la propagation plus ou moins importante des erreurs d'arrondi dans un calcul donné. Nous les rencontrerons à différents endroits dans ce cours, notamment :

– la stabilité d'un schéma de résolution d'équations différentielles, voir Définition 4.2
– le conditionnement de la résolution d'un système linéaire, voir la section correspondante dans le chapitre sur la résolution de systèmes.
– la stabilité d'un schéma de résolution d'une équation aux dérivées partielles par différence finie, voir section 3.2.3.3

Nous les étudions ici dans le cas plus simple de l'évaluation d'une fonction.

$$x \in \mathbb{R} \longmapsto f(x) \in \mathbb{R}.$$

1.3.1 Conditionnement

Le conditionnement d'une fonction décrit la sensibilité de la valeur de cette fonction à une petite variation de son argument

$$\frac{f(x) - f(x*)}{f(x)} \text{ en fonction de } \frac{x - x*}{x}$$

lorsque $x - x*$ est petit. Pour une matrice A, son conditionnement est un nombre qui mesure la variation relative de la solution du système linéaire $AX = b$ quand on perturbe le second membre b ou les coefficients de la matrice eux-mêmes. Plus généralement, on dira qu'un problème est *bien conditionné* si de petites variations sur les données induisent de petites variations sur le résultat. Dans le cas contraire, on dit qu'il est mal conditionné.

Pour en revenir au conditionnement d'une fonction, si celle-ci est suffisamment régulière, mettons de classe C^1, on a évidemment :

$$\left| \frac{f(x) - f(x*)}{f(x)} \Big/ \frac{x - x*}{x} \right| \simeq \left| \frac{xf'(x)}{f(x)} \right|.$$

D'où on tire :

Définition 1.1 *On appelle conditionnement d'une fonction numérique f de classe C^1 en un point x, le nombre*

$$cond(f)_x := \left| \frac{xf'(x)}{f(x)} \right|.$$

Exemple 1.4 : $f(x) = \sqrt{x}$

$$\left| \frac{xf'(x)}{f(x)} \right| = \frac{1}{2}.$$

Ceci correspond à un bon conditionnement, puisque l'erreur relative sur $f(x)$ sera au plus moitié d'une erreur relative sur x.

Exemple 1.5 : $f(x) = a - x$

$$\left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x}{a - x} \right|.$$

Ici, le conditionnement est très mauvais si x est voisin de a. Ceci correspond aux exemples 1.1, 1.2, 1.3.

1.3.2 Stabilité

La stabilité décrit la sensibilité d'un algorithme numérique pour le calcul d'une fonction $f(x)$. Plus généralement la stabilité d'un algorithme décrira sa sensibilité à la propagation des erreurs. C'est donc une notion extrêmement importante en analyse numérique et nous la retrouverons à différentes reprises dans ce cours.

Exemple 1.6 : Revenons au cas d'une fonction et considérons

$$f(x) = \sqrt{x+1} - \sqrt{x}.$$

Le conditionnement de cette fonction est égal à :

$$\left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\left(\sqrt{x} - \sqrt{x+1}\right)}{2\sqrt{x(x+1)}} - \frac{1}{\sqrt{x+1}-\sqrt{x}}\right| = \frac{1}{2}\sqrt{\frac{x}{x+1}}$$

Cette dernière expression étant proche de $\frac{1}{2}$ pour x grand. Donc, si x est grand, le conditionnement de f est bon. Cependant, dans un calcul à 6 chiffres significatifs, on a :

$$f(12345) = \sqrt{12346} - \sqrt{12345} = 111,113 - 111,108 = 0,500000.10^{-2}.$$

Or un calcul précis donne : $f(12345) = 0,4500032....10^{-2}$. On a donc une erreur de 10% ce qui est très important et peu en accord avec le bon conditionnement de f. Ceci est dû à l'algorithme utilisé dans ce calcul que l'on peut expliciter comme suit :

$$\begin{aligned} x_0 &: = 12345 \\ x_1 &: = x_0 + 1 \\ x_2 &: = \sqrt{x_1} \\ x_3 &: = \sqrt{x_0} \\ x_4 &: = x_2 - x_3 \end{aligned}$$

Il y a quatre fonctions à intervenir et, à priori, même si le conditionnement de f est bon, il se peut que le conditionnement d'une ou plusieurs fonctions utilisées dans l'algorithme soit supérieur à celui de f. C'est ce qui se produit ici pour la fonction $x_3 \longmapsto x_4 = x_2 - x_3$ (x_2 étant supposé fixe) dont le conditionnement est grand lorsque x_3 est voisin de x_2 comme on l'a vu précédemment.
En conclusion, le choix d'un bon algorithme numérique est essentiel. Par exemple, ci-dessus, un meilleur algorithme est obtenu en utilisant :

$$f(x) = \sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1}+\sqrt{x}}.$$

Dans ce cas, toujours avec 6 chiffres significatifs :

$$f(12345) = \frac{1}{\sqrt{12346}+\sqrt{12345}} = \frac{1}{222,221} = 0,450002.10^{-2}$$

ce qui donne une erreur relative de $0,0003\%$. Dans la même optique, l'exemple suivant peut aussi être médité.

Exemple d'instabilité : Imaginons qu'on souhaite calculer e^x à l'aide de sa série de Taylor

$$e^x \simeq \sum_{n=0}^{N} \frac{x^n}{n!} \text{ pour } N \text{ grand}$$

ce qui est naturel car on sait que la série est très rapidement convergente. On peut alors écrire le programme Matlab suivant (on somme jusqu'à ce que le nouveau terme $x^n/n!$ qu'on ajoute ne change plus rien) :

```
function z=exptay(x)\\z=1 ;
zold=0 ;
y=1 ;
n=0 ;
while z ~ = zold
n=n+1 ;
y=y*x/n ;
zold=z ;
z=z+y ;
end
```

Pour $x = -21$ le programme précédent donne la valeur (négative !!) $-3.1649 * 10^{-9}$ alors que la valeur exacte est $\exp(-21) \simeq 0.7582560 * 10^{-9}$. La raison en est que pour certaines valeurs de n dans l'algorithme on somme des nombres très grands, par exemple pour $n = 19, 20, 21, 22$ le terme $y = (-21)^n/n!$ est supérieur à 10^8. Donc l'erreur de chute qu'on commet à ce moment là, même en travaillant avec 16 chiffres significatifs, pollue complètement le résultat final. Il s'avère que pour $x > 0$, la formule $e^{-x} = \frac{1}{e^x}$ donne lieu à un calcul beaucoup plus stable, même si e^x est calculé avec la formule de Taylor.

1.3.2.1 Algorithme implicite

Donnons un dernier exemple [1]. Supposons qu'on souhaite calculer, pour les premières valeurs de l'entier n, l'intégrale

$$I_n = \int_0^1 \frac{x^n}{x+10}\, dx.$$

On obtient facilement $0 < I_n < I_{n+1} < 1/10(n+2)$ pour tout n. Il semble astucieux de calculer I_n par récurrence puisqu'une simple intégration par parties donne

$$I_{n+1} = \frac{1}{n+1} - 10 I_n. \tag{1.1}$$

En partant alors de $I_0 = \log 1.1$ et en calculant les différents termes de la suite par la formule de récurrence (1.1) on obtient (en supposant qu'on travaille avec 5 chiffres après

1. Exemple adapté de Dahlquist-Bjorck [1974]

la virgule) :

$$\begin{aligned} I_0 &\simeq 0.09531 \\ I_1 &\simeq 1 - 0.9531 = 0.0469 \\ I_2 &\simeq 0.5 - 0.469 = 0.0310 \\ I_3 &\simeq 0.33333 - 0.31000 = 0.02333 \\ I_4 &\simeq 0.25 - 0.2333 = 0.0167 \\ I_5 &\simeq 0.2 - 0.167 = 0.033 \\ I_6 &\simeq 0.16667 - 0.33 = -0.16333 \\ I_7 &\simeq 0.125 + 1.6333 = 1.7583 \\ I_8 &\simeq 0.11111 - 17.583 = -17.47189 \end{aligned}$$

On voit que dès le calcul de I_5 il y a un problème car la suite n'est plus décroissante. Ensuite les valeurs donnent franchement n'importe quoi. On a là un phénomène d'instabilité typique car les valeurs calculées commencent à changer de signe avec une valeur absolue qui grandit de plus en plus. En fait, on en comprend facilement la cause : le premier terme I_0 est forcément connu avec une erreur ε (inférieure à 10^{-6} ici), mais la multiplication par 10 à chaque étape de la formule de récurrence multiplie également l'erreur par 10. Aussi dès le 6ème terme, on peut imaginer que l'erreur est de l'ordre de $10^6\varepsilon$ donc de l'ordre de l'unité et c'est bien ce qu'on constate.

Il serait beaucoup plus intelligent ici d'utiliser la formule de récurrence en sens inverse :

$$I_n = \frac{1}{10(n+1)} - \frac{I_{n+1}}{10} \tag{1.2}$$

c'est-à-dire de façon **implicite** (puisqu'on suppose qu'on connait I_{n+1} au moment où on veut calculer I_n). En effet, on sent bien que cette fois l'erreur sera divisée par 10 (et non multipliée par 10) à chaque itération. Donc même en partant avec un résultat très faux, du type $I_{15} = 0$, on va avoir $I_9, I_8, \ldots$ avec une excellente précision :

$$\begin{aligned} I_{14} &\simeq 0.00667 \\ I_{13} &\simeq 0.00714 - 0.00066 = 0.00648 \\ I_{12} &\simeq 0.00769 - 0.00065 = 0.00704 \\ I_{11} &\simeq 0.00833 - 0.00070 = 0.00763 \\ I_{10} &\simeq 0.00909 - 0.00076 = 0.00833 \\ I_9 &\simeq 0.01 - 0.00083 = 0.00917 \\ I_8 &\simeq 0.01111 - 0.00092 = 0.01019 \\ I_7 &\simeq 0.0125 - 0.00102 = 0.01148 \\ I_6 &\simeq 0.01429 - 0.00115 = 0.01314 \end{aligned}$$

en fait, les résultats sont exacts, dès $n = 12$ (et donc pour toutes les valeurs $n \leq 12$).
Nous verrons d'autres exemples dans la suite où le choix d'un algorithme implicite permet d'éviter de gros problèmes d'instabilité.

1.4 Conclusion

Afin de limiter la propagation des erreurs d'arrondi, il faut essayer d'anticiper en utilisant des algorithmes dont la stabilité est optimisée par un choix d'opérations intermédiaires à bon conditionnement.

Les phénomènes soulignés dans ce chapitre ne pouvant être, en tout état de cause, complètement éliminés, on peut essayer d'évaluer l'erreur totale à laquelle un algorithme est susceptible de donner lieu :
a/ en faisant un calcul en double précision et en confrontant le résultat au même calcul fait en simple précision. C'est ce que nous avons fait dans l'exemple 1.2. Cependant, cette technique est très coûteuse en temps machine puisqu'elle peut multiplier le temps de calcul par un facteur 8.
b/ en faisant une analyse mathématique de l'erreur : ce peut être une analyse rétrograde de l'erreur comme celle utilisée plus haut pour comparer $(a+b)+c$ et $a+(b+c)$. Des méthodes statistiques peuvent être également utilisées. Nous renvoyons à la littérature spécialisée pour ces questions le plus souvent délicates.

1.5 Exercices du chapitre 1

Exercice 1.1 En écrivant un petit programme, trouver la capacité et le pas de votre calculatrice de poche ou de l'ordinateur.

Exercice 1.2 Calculer les racines de l'équation $x^2 + 111,11x + 1,2121 = 0$ dans une arithmétique à 5 chiffres significatifs (base 10) en utilisant les formules $x = \frac{-b+\sqrt{b^2-4ac}}{2a}$, puis $x = \frac{-2c}{b+\sqrt{b^2-4ac}}$, et analyser les résultats.

Exercice 1.3 Estimer l'erreur faite dans le calcul de $(\cos x)\, e^{10x^2}$ autour de $x = 2$ quand l'erreur sur x est d'ordre 10^{-6}.

Exercice 1.4 Trouver une méthode pour calculer :

$$\sin(\alpha + x) - \sin\alpha$$

lorsque x est voisin de 0, à la précision permise par le nombre de chiffres significatifs utilisés.

Exercice 1.5 Calculer, en arrondissant à quatre chiffres significatifs chaque calcul intermédiaire, la somme des nombres suivants d'abord dans le sens croissant puis dans le sens décroissant. Comparer les résultats.

$$0,1580 \qquad 0,2653 \qquad 0,2581.10^1 \qquad 0,4288.10^1 \qquad 0,6266.10^2 \qquad 0,7555.10^2$$

$$0,7889.10^3 \qquad 0,7767.10^3 \qquad 0,8999.10^4.$$

Exercice 1.6 Les valeurs 1, $\frac{1}{6}$, $\frac{1}{6^2}$, ..., $\frac{1}{6^n}$ sont théoriquement générées par la suite $x_0 = 1$, $x_1 = \frac{1}{6}$, $x_{n+1} = \dfrac{37}{6}x_n - x_{n-1}$ pour tout $n \geq 1$.
Calculer les valeurs successives avec 4 décimales puis avec 8 décimales et comparer... On constate une différence très importante dès la 5ème étape. Expliquer !

1.6 TD1 : Initiation Matlab

1. Découverte de l'espace de travail

2. Aide

2.1 Que fait la fonction Matlab *rem* ?
2.2 Comment calculer les racines d'un polynôme à l'aide de Matlab ? Quel algorithme est utilisé ?
2.3 Comment calculer numériquement une intégrale avec Matlab ?

3. Constantes prédéfinies

3.1 Combien vaut la constante *realmax* ?
3.2 Combien vaut *2*realmax* ?
3.3 Combien vaut *realmax+1* ? Pourquoi ?

4. Les matrices

4.1 Définir la matrice A suivante :

$$A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

4.2 Extraire l'élément a_{22}, la première ligne, la deuxième colonne.
4.3 Définir les vecteurs suivants :

$$V1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \text{ et } V2 = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

Que vaut le produit de A avec $V1$? Avec $V2$?
4.4 Que fait Matlab avec les expressions *V1*V2*, *V2*V1*, *V1.*V2*, *V1.*V2'* ?
4.5 Construire la matrice B définie par blocs :

$$B = \left(\begin{array}{c|c} A & -A \\ \hline 2A & A \end{array}\right)$$

4.6 Définir la matrice 8*8 suivante (utiliser la fonction *diag*) :

$$D = \begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & 2 & 1 \\ & & & 1 & 2 \end{pmatrix}$$

Résoudre le système $Dx = y$ où $y^t = (9\ 9\ \cdots\ 9)$.

5. Les graphiques

6.1 Tracer la courbe représentative de la fonction $f : x \mapsto Arcsin(cos(x))$ pour x compris entre -4π et 4π en pointillé et en rouge. Ajouter sur le même graphe des croix bleues aux points $(x_k, f(x_k))$, avec $x_k = k\pi$, $-4 \leq k \leq 4$.

6.2 Tracer la courbe d'équations paramétriques :

$$\begin{cases} x(t) = \cos(t) + t\sin(t) \\ y(t) = \sin(t) - t\cos(t) \end{cases} \quad \text{avec} \quad -15 \leq t \leq 15$$

Modifier l'aspect de la figure pour que l'axe des abscisses et l'axe des ordonnées aient la même échelle.

6.3 Tracer la surface d'équation : $z = \sin(x + \sin y)$ avec $-3 \leq x \leq 3$ et $-3 \leq y \leq 3$.

6. Programmation

7.1 Ecrire une fonction *triangle.m* qui, étant donné trois nombres a, b et c, renvoie le périmètre et l'aire du triangle dont la longueur des cotés est égale à a, b et c. On rappelle que si P est le périmètre, alors la surface est donné par :

$$S = \frac{1}{4}\sqrt{P(P-2a)(P-2b)(P-2c)}$$

7.2 Modifier cette fonction en une fonction *triangle2.m* pour la rendre "vectorielle". C'est à dire qu'elle puisse prendre non pas trois réels en entrée mais trois vecteurs de réels de même dimension A, B et C et calculer l'aire et le périmètre de tous les triangles $(A(i), B(i), C(i))$.

7.3 Modifier la fonction *triangle.m* pour que, s'il n'existe aucun triangle dont les longueurs des cotés soient a, b et c, alors l'exécution de la fonction soit stoppée et un message d'erreur affiché pour l'utilisateur.

7.4 Que calcule Matlab avec les deux fonctions :

function [a1,t1]=suite1(n)
tic ;
a=zeros(1,n) ;a(1)=1 ;a(2)=1 ;
for k=3 :n
a(k)=a(k-1)+a(k-2) ;
end
a1=a(n) ;
t1=toc ;

function [a2,t2]=suite2(n)
tic ;
A=[1 1 ;1 0] ;
X=A^(n-2)*[1 ;1] ;
a2=X(1) ;
t2=toc ;

Comparer les temps de calcul.

7. Fonction Matlab

8.1 Quel est le minimum de la fonction $p : x \mapsto \frac{1}{2}x^2 - \frac{1}{4}\ln x$?

8.2 Donner une valeur approchée de l'intégrale suivante :

$$\int_0^1 \frac{e^{-x}}{1+x^2}dx$$

8.3 Résoudre numériquement l'équation :

$$\cos x = e^{-x}$$

1.7 Aide-mémoire

1.7.1 Formules de Taylor

Il existe différentes formules de Taylor qui diffèrent essentiellement par l'expression du "reste".

Pour une fonction f de classe C^n (c'est-à-dire une fonction n fois dérivable et dont la n-ième dérivée est une fonction continue), on a la :
Formule de Taylor-Young :

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + h^n \varepsilon(h) \tag{1.3}$$

où $\varepsilon(h)$ est une fonction qui tend vers 0 quand h tend vers 0.

Si on veut une expression plus précise du reste, on a les deux formules suivantes. Pour une fonction f de classe C^{n+1}, on a la :
Formule de Taylor-Lagrange :

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \frac{h^{n+1}}{(n+1)!} f^{(n+1)}(x+\theta h) \tag{1.4}$$

où θ est un nombre (qui dépend de x et de h) compris entre 0 et 1.
On a aussi la
Formule de Taylor avec reste intégral :

$$f(x+h) = f(x)+hf'(x)+\frac{h^2}{2} f''(x)+\ldots+\frac{h^n}{n!} f^{(n)}(x)+\int_x^{x+h} \frac{(x+h-t)^n}{n!} f^{(n+1)}(t)\, dt \tag{1.5}$$

Enfin, on parle de série de Taylor quand on fait tendre n vers $+\infty$ dans les formules précédentes, tout en s'assurant de la convergence de la série par les techniques usuelles.

1.7.2 Intégration par parties

Si f et g sont des fonctions de classe C^1 sur l'intervalle $[a, b]$, on a la classique
Formule d'intégration par parties :

$$\int_a^b f'(t)g(t)\, dt = (f(b)\, g(b) - f(a)\, g(a)) - \int_a^b f(t)g'(t)\, dt \tag{1.6}$$

Chapitre 2

Interpolation et approximation polynômiale

Dans ce chapitre, on dispose d'une fonction f, connue par exemple uniquement par ses valeurs en certains points, et on cherche à remplacer ou à approcher f par une fonction plus simple, le plus souvent par un polynôme. Nous verrons dans ce contexte, *l'interpolation* qui consiste à rechercher un polynôme qui passe exactement par les points donnés, *l'interpolation par morceaux*, en particulier *les fonctions splines* où l'expression du polynôme est différente sur chaque sous-intervalle, *l'approximation au sens des moindres carrés* ou on cherche à approcher au mieux la fonction, pour la norme euclidienne. Cette dernière approche conduira naturellement aux fonctions trigonométriques et aux séries de Fourier, et on présentera très rapidement la *Transformée de Fourier Rapide* ou F.F.T.

2.1 Interpolation de Lagrange

2.1.1 Le polynôme d'interpolation

Soit $f : [a, b] \to \mathbb{R}$ connue en n+1 points distincts $x_0, x_1, ...x_n$ de l'intervalle $[a, b]$. Il s'agit de construire un polynôme P de degré inférieur ou égal à n tel que

$$\forall i = 0, 1, ...n \qquad P(x_i) = f(x_i) \tag{2.1}$$

Théorème 2.1 *Il existe un et un seul polynôme de degré inférieur ou égal à n solution de (2.1). Le polynôme s'écrit*

$$P_n(x) = \sum_{i=0}^{n} f(x_i)\, L_i(x) \tag{2.2}$$

où

$$L_i(x) = \prod_{k=0 \quad k\neq i} \frac{(x - x_k)}{(x_i - x_k)}. \tag{2.3}$$

Remarque 2.1 Le polynôme P_n est appelé polynôme d'interpolation de Lagrange de la fonction f aux points $x_0, x_1, ...x_n$.
Les polynômes $L_i(x)$ sont appelés polynômes de base de Lagrange associés à ces points.

Démonstration du Théorème 2.1 :
Existence : On vérifie directement que le polynôme donné par (2.2) est solution de (2.1). Pour cela, on utilise le fait que $L_i(x_j) = \delta_{ij}$ (c'est-à-dire 1 si $i = j$, 0 sinon).
Unicité : Soit Q un autre polynôme solution. Alors $\forall i = 0, 1, ...n \;\; Q(x_i) - P(x_i) = 0$. Ainsi $Q - P$ est un polynôme de degré inférieur ou égal à n s'annulant en $n + 1$ points. Il est donc identiquement nul.

Exemple 2.1 Interpolation linéaire. On applique (2.2) avec $n = 1$ pour trouver

$$P_1(x) = f(x_0)\frac{(x - x_1)}{(x_0 - x_1)} + f(x_1)\frac{(x - x_0)}{(x_1 - x_0)}.$$

Ceci s'écrit encore

$$P_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0). \tag{2.4}$$

Remarque 2.2 L'écriture (2.2) du polynôme d'interpolation est intéressante au point de vue théorique, mais peu du point de vue numérique : elle a un caractère peu algorithmique. De plus, son évaluation requiert trop d'opérations élémentaires.

On lui préfère la formule de Newton qui consiste à écrire le polynôme d'interpolation P_n aux points $x_0, x_1, ...x_n$ sous une forme généralisant (2.4) soit :

$$P_n(x) = a_0^n + a_1^n(x - x_0) + ... + a_n^n(x - x_0)(x - x_1)...(x - x_{n-1}).$$

Notons d'abord que tout polynôme de degré inférieur ou égal à n peut se mettre (et ce de manière unique) sous cette forme dès que les x_i sont tous distincts, car la famille $1, x - x_0, \ldots, (x - x_0)(x - x_1)...(x - x_{n-1})$ est une base de l'espace vectoriel des polynômes de degré inférieur ou égal à n. L'avantage de cette écriture est que la partie tronquée

$$a_0^n + a_1^n(x - x_0) + a_2^n(x - x_0)(x - x_1) + ... + a_{n-1}^n(x - x_0)(x - x_1)...(x - x_{n-2})$$

n'est pas autre chose que $P_{n-1}(x)$: en effet, il s'agit d'un polynôme de degré inférieur ou égal à $n - 1$ tel que :

$$\forall i = 0, 1, ...n - 1 \qquad \text{Valeur en } \; x_i = P_n(x_i) = f(x_i).$$

Donc, connaissant P_{n-1}, il suffit de calculer a_n^n pour connaître P_n (a_n^n est aussi le coefficient de x^n dans P_n écrit sous forme usuelle). Les a_n^i sont donnés par la formule de Newton :

Théorème 2.2 (Formule d'interpolation de Newton) *Le polynôme d'interpolation de Lagrange de la fonction f aux points distincts $x_0, x_1, ...x_n$ est donné par :*

$$P_n(x) = \sum_{i=0}^{n} f[x_0, x_1, ...x_i] \prod_{k=0}^{i-1}(x - x_k) \tag{2.5}$$

où $f[.]$ désigne les différences divisées de f définies par la formule de récurrence :

$$i = 0, ..., n \qquad f[x_i] = f(x_i) \tag{2.6}$$

$$f[x_0, x_1, ..., x_k] := \frac{f[x_1, ..., x_k] - f[x_0, ..., x_{k-1}]}{x_k - x_0}. \tag{2.7}$$

Remarque 2.3 Par convention $\prod_{k=0}^{i-1}(x-x_k) := 1$ si $i < 1$.

Démonstration : On la fait par récurrence sur le nombre de points x_i.
– La formule (2.5) est vraie pour $n = 1$ (cf. (2.4)).
– Supposons la vraie pour n points. On a alors :

$$P_n(x) = P_{n-1}(x) + a_n^n \prod_{k=0}^{n-1}(x-x_k) = \sum_{i=0}^{n-1} f\left[x_0, x_1, ..., x_i\right] \prod_{k=0}^{n-2}(x-x_k) + a_n^n \prod_{k=0}^{n-1}(x-x_k).$$

Reste à calculer a_n^n. Considérons

$$Q(x) = \frac{x-x_0}{x_n-x_0} Q_{n-1}(x) + \frac{x_n-x}{x_n-x_0} P_{n-1}(x)$$

où $Q_{n-1}(x) = \sum_{i=1}^{n} f\left[x_1, ..., x_i\right] \prod_{k=1}^{n-1}(x-x_k)$ est le polynôme d'interpolation de f aux n points $x_1, x_2, ..., x_n$. On vérifie immédiatement que :

$$\forall i = 0, 1, ..., n \qquad Q(x_i) = f(x_i).$$

Puisque $\deg Q \leq n$, $Q \equiv P_n$. Cette nouvelle expression de P_n nous permet d'affirmer que le coefficient de x^n de P_n est donné par

$$a_n^n = \frac{1}{x_n - x_0} f\left[x_1, ..., x_n\right] - \frac{1}{x_n - x_0} f\left[x_0, x_1, ..., x_{n-1}\right].$$

2.1.2 Propriétés des différences divisées

i) $f\left[x_0, x_1, ..., x_n\right]$ est invariant par des permutations sur les x_i : en effet, permuter les x_i ne change pas le polynôme d'interpolation et donc ne change pas le coefficient du terme de plus haut degré.
ii) si $f = Q$ est un polynôme de degré q

$$Q\left[x_0, x_1, ..., x_n\right] = \begin{cases} 0 \text{ si } q < n \\ \text{coefficient du terme de plus haut degré de } Q \text{ si } q = n. \end{cases}$$

En effet, si $q \leq n$, l'interpolé de f n'est autre que Q lui-même.
iii) table des différences divisées : permet de calculer de façon inductive les différences divisées d'une fonction selon le schéma suivant :

$$\begin{array}{cccccc}
x_0 & f[x_0] & & & & \\
 & & f[x_0,x_1] & & & \\
x_1 & f[x_1] & & f[x_0,x_1,x_2] & & \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] & \\
x_2 & f[x_2] & & f[x_1,x_2,x_3] & & \ddots \\
 & & f[x_2,x_3] & & & \\
x_3 & f[x_3] & & & & \\
\vdots & \vdots & & \vdots & & \\
\vdots & \vdots & f[x_{n-1},x_n] & & & \\
x_n & f[x_n] & & & &
\end{array}$$

Les coefficients de la formule de Newton sont donnés par les premiers éléments de chaque colonne (diagonale (1)).

iv) Interpolation entre des points équidistants : Les différences divisées peuvent alors être remplacées par des différences finies. On note :

$$\begin{aligned} &x_i = a + ih, \quad i = 0, ..., N, \quad N = \tfrac{a-b}{h} \\ &s := s(x) = \tfrac{x-x_0}{h} \Longleftrightarrow x = x_0 + sh \\ &f(x) = f(x_0 + sh) = f_s. \end{aligned}$$

Les différences finies sont obtenues par la formule de récurrence

$$\Delta^i f_s = \begin{cases} f_s \text{ si } i = 0 \\ \Delta^{i-1} f_{s+1} - \Delta^{i-1} f_s \text{ si } i > 0 \end{cases}$$

On note aussi $\Delta^1 = \Delta$. Le lien avec les différences finies est :

$$f[x_k, ..., x_{k+i}] = \frac{1}{i! h^i} \Delta^i f_k. \tag{2.8}$$

Ceci se démontre facilement par récurrence sur le nombre de points. La formule d'interpolation de Lagrange devient alors, (au lieu de (2.5)) ce qu'on appelle la formule de Newton progressive :

$$P_n(x) = P_n(x_0 + sh) = \sum_{i=0}^{n} \binom{s}{i} \Delta^i f_0 \tag{2.9}$$

où on pose

$$\binom{s}{i} = \begin{cases} \dfrac{s(s-1)...(s-i+1)}{1.2...i} \text{ si } i > 0 \\ 1 \text{ si } i = 0 \end{cases} \tag{2.10}$$

Remarque 2.4 La notation (2.10) est bien sûr une généralisation de la notation utilisée pour les coefficients binomiaux correspondants à s entier positif (=nombre de façons de choisir i éléments parmi s).

La formule (2.9) se démontre à partir de (2.5) et de (2.8) : on a en effet, en utilisant aussi $x - x_k = (s-k)h$

$$P_n(x) = \sum_{i=0}^{n} \frac{1}{i! h^i} \Delta^i f_0 . h^i \prod_{k=0}^{i-1} (s-k).$$

Remarque 2.5 avant l'ère des ordinateurs, ces formules étaient utilisées pour calculer des valeurs approchées d'une fonction f en tout point à partir des valeurs données dans une table (généralement en des points équidistants). le calcul des différences finies successives permettait également de détecter les erreurs dans les tables. En effet, même de faibles erreurs sur les valeurs de f peuvent entraîner des oscillations importantes des différences finies d'ordre supérieur.

2.1.3 Erreur dans l'interpolation de Lagrange

Le but de l'interpolation étant de remplacer l'évaluation de $f(x)$ par celle de $P_n(x)$, il est important de connaître l'erreur

$$E_n(x) = f(x) - P_n(x), \quad x \in [a, b]$$

Exemple 2.2 Soit $f(x) = \dfrac{1}{1+x^2}$: pour une interpolation sur l'intervalle $[-5, 5]$ avec des points équidistants,

$$x_i = i\frac{10}{n} - 5, \quad i = 0, 1, ..., n$$

on obtient les résultats suivants : n est le nombre de points et

$$E_n = \max_{-5 \le x \le 5} [f(x) - P_n(x)] \approx \max_i |f(y_i) - P_n(y_i)|$$

où on prend $y_i = \frac{i}{10} - 5, i = 0, ..., 100$.

n	2	4	6	8	10	12	14	16
E_n	0.6	0.4	0.6	1.05	1.91	3.6	7.2	14.0

Dans cet exemple, l'erreur augmente très vite avec n ... ! Ceci est bien sûr paradoxal dans la mesure où on utilise de plus en plus d'informations sur f ! On comprendra mieux ce qui se passe après avoir vu le Théorème d'erreur suivant.

Théorème 2.3 *Soit $f : [a, b] \longrightarrow \mathbb{R}$, $n+1$ fois continument différentiable et P_n le polynôme d'interpolation de Lagrange aux points $x_0, x_1, ..., x_n$ de $[a, b]$. Alors*

$$|f(x) - P_n(x)| \le \frac{M_{n+1}}{(n+1)!} |\pi_n(x)| \tag{2.11}$$

où

$$M_{n+1} = \max_{a \le x \le b} \left| f^{(n+1)}(x) \right| \tag{2.12}$$

et

$$\pi_n(x) = \prod_{i=0}^{n} (x - x_i). \tag{2.13}$$

La démonstration va reposer sur le Lemme suivant.

Lemme 2.1 *Sous les hypothèses du théorème, il existe $\zeta \in [a, b]$ tel que*

$$f[x_0, ..., x_n] = \frac{f^{(n)}(\zeta)}{n!}.$$

Démonstration : La fonction $E_n(x) = f(x) - P_n(x)$ s'annule en $n+1$ points distincts. D'après le théorème de Rolle [1], $E_n'(x)$ s'annule en au moins n points distincts de $[a, b]$. A nouveau, d'après le même théorème, $E_n''(x)$ s'annule $n-1$ fois, etc... On obtient ainsi que $E_n^{(n)}$ s'annule en un point $\zeta \in [a, b]$, soit $f^{(n)}(\zeta) = P_n^{(n)}(\zeta)$. Comme P_n est de degré n,

1. voir l'Aide-mémoire, section 2.6

$P_n^{(n)}(\zeta) = a_n n!$ où a_n est le coefficient de x_n, soit ici $a_n = f[x_0, x_1, ..., x_n]$. Ce qui démontre le lemme.

Démonstration du théorème : Soit $\overline{x} \in [a,b]$; notons Q le polynôme d'interpolation aux points $x_0, x_1, ..., x_n, \overline{x}$. D'après la formule de Newton, il est donné au point $\overline{x}$ par

$$Q(\overline{x}) = P_n(\overline{x}) + f[x_0, x_1, ..., x_n, \overline{x}]\,\pi_n(\overline{x}).$$

Mais puique $Q(\overline{x}) = f(\overline{x})$ ceci donne

$$f(\overline{x}) - P_n(\overline{x}) = f[x_0, x_1, ..., x_n, \overline{x}]\,\pi_n(\overline{x}). \tag{2.14}$$

Il suffit alors d'appliquer le lemme pour conclure à (2.11).

2.1.3.1 Analyse du Théorème 2.3

Exemple 2.3 Soit $f(x) = e^x$; pour $x \in [a,b]$, on a $M_{n+1} = e^b$. D'autre part, on a la majoration facile $|\pi_n(x)| \leq (b-a)^{n+1}$ (pour tout choix des $n+1$ points x_i). Donc, d'après l'estimation (2.11) :

$$\forall x \in [a,b], \quad |f(x) - P_n(x)| \leq \frac{(b-a)^{n+1}}{(n+1)!} e^b.$$

En particulier,

$$\lim_{n\to\infty} |f(x) - P_n(x)| = 0.$$

Donc, dans le cas de la fonction exponentielle, plus on prend de points, meilleure est l'interpolation.

Remarque 2.6 On peut en fait démontrer la même chose pour toute fonction développable en série entière au point $\frac{a+b}{2}$ avec un rayon de convergence $r > \frac{3}{2}(b-a)$.

Lorsqu'on prend des points x_i équidistants dans $[a,b]$ - ce qui est fréquemment utilisé vu la simplicité des calculs - on peut montrer que

$$\max_{a\leq x\leq b} |\pi_n(x)| \leq C \frac{e^{-n}}{\sqrt{n}\log n} (b-a)^{n+1} \tag{2.15}$$

où C est une constante positive. Analysons l'exemple 2.2, à l'aide de ce résultat.
Soit $f(x) = \frac{1}{1+x^2}, a = -b > 0$. Pour dériver facilement f, on remarque que $f(x) = \Im m\left(\frac{1}{x-i}\right)$, d'où $f^{(k)}(x) = \Im m\left[\frac{(-1)^k}{(x-i)^{k+1}}\right] k!$; on en déduit, puisque

$$\frac{1}{(x-i)^{k+1}} = \left(\frac{x+i}{x^2+1}\right)^{k+1} = \frac{1}{(x^2+1)^{\frac{k+1}{2}}} (\cos\theta + i\sin\theta)^{k+1}$$

où θ est tel que $\cos\theta = \frac{x}{\sqrt{x^2+1}}, \sin\theta = \frac{1}{\sqrt{x^2+1}}$

$$f^{(k)}(x) = \frac{(-1)^k k!}{(x^2+1)^{\frac{k+1}{2}}} \sin(\theta(k+1)).$$

On constate que, au moins pour k grand, $M_k \sim k!$. Les formules (2.11), (2.12) et (2.15) donnent donc :

$$\max_{-a\leq x\leq a} |E_n(x)| < C\frac{e^{-n}}{\sqrt{n}\log n}(2a)^{n+1}.$$

Conclusion : Si $2a < e$, $E_n(x)$ converge vers 0 avec n.

Dans l'exemple 3.1, $a = 5$. Dans ce cas, le majorant ci-dessus tend vers l'infini. Ceci ne permet plus de rien affirmer sur $E_n(x)$. Le calcul numérique de l'exemple 3.1 suggère que E_n augmente au lieu de diminuer. On peut en fait montrer que $\max_{|x|\leq 5} |E_n(x)|$ ne tend pas vers 0. Il faut donc être très prudent avec l'interpolation polynômiale.

L'estimation (2.11) du Théorème 2.3 montre que l'erreur dépend d'une part des dérivées de f, d'autre part du maximum de la fonction π_n, qui lui ne dépend que du choix des x_i. On peut chercher à minimiser $\max_{a\leq x\leq b} |\pi_n(x)|$ en choisissant au mieux les points x_i.

Il se trouve qu'un choix de points équidistants n'est pas du tout optimal, loin de là. On peut démontrer que le choix optimal est obtenu à l'aide des points de Chebyshev, soit $x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{(2i+1)\pi}{2n+2}, i = 0, 1, ..., n$.

Les graphes de $|\pi_n(x)|$ pour un choix de points équidistants et pour les points de Chebyshev sont présentés à la figure 2.1

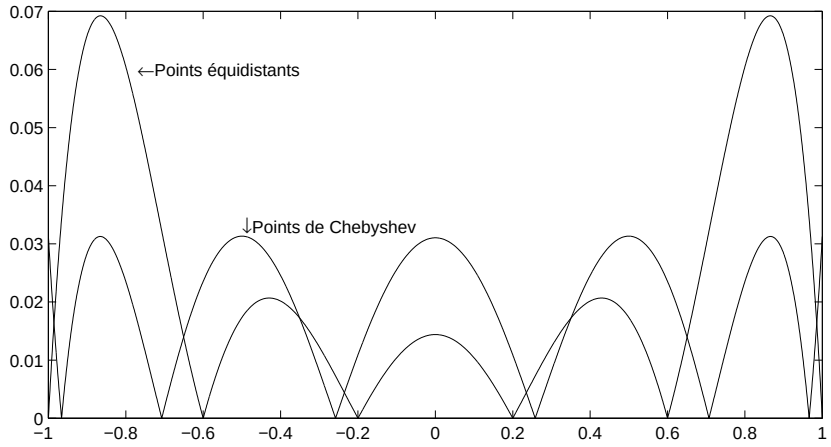

FIGURE 2.1 – Points de Chebyshev et points équidistants

On voit que, pour des points équidistants, $|\pi_n(x)|$ est relativement grand au voisinage des extrémités. Si on se limite à des points équidistants, il vaudra donc mieux se placer à l'intérieur de l'intervalle et surtout éviter les extrémités. On remarque que, pour les points de Chebyshev, le maximum de $|\pi_n(x)|$ est nettement plus petit. D'autre part, tous les maximums relatifs sont égaux : il semble bien que ce choix "régularise" les différents pics en les uniformisant.

2.2 Approximation au sens des moindres carrés

2.2.1 Introduction

La méthode que nous allons voir maintenant s'appelle en statistiques *ajustement des données ou régression linéaire*. Jusque ici, nous avons considéré l'approximation d'une fonction par un procédé d'interpolation à l'aide des valeurs $f(x)$ de f en des points x_k. Ceci présuppose que ces valeurs soient connues, et ce de façon assez précise. Il y a beaucoup de situations où ce n'est pas le cas en particulier lorsque les valeurs $f(x)$ proviennent de mesures physiques.

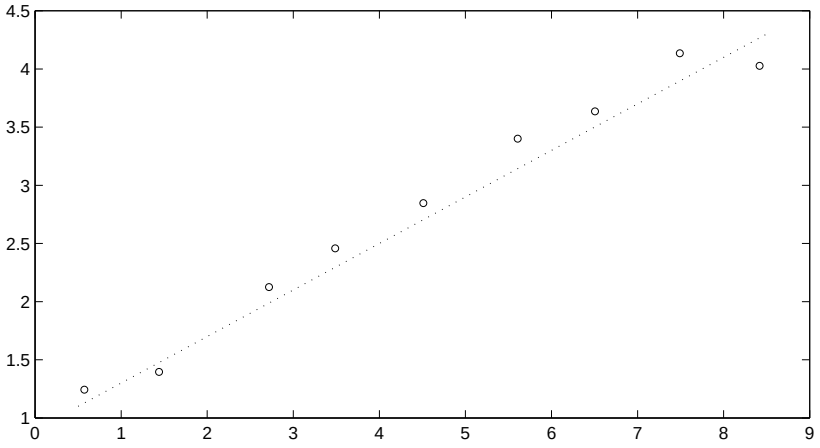

FIGURE 2.2 – Droite des moindres carrés

Exemple 2.4 Le résultat de mesures physiques tel que ci-dessus conduit à penser que $f(x)$ doit être une fonction linéaire. Il ne serait pas très raisonnable de remplacer $f(x)$ par son polynôme d'interpolation en les points x_k dont le calcul dépendrait uniquement des valeurs manifestement erronées $f(x_k)$.

Une analyse succinte du phénomène ci-dessus, fréquent dans la réalité physique, conduit à penser que ces valeurs mesurées $f(x_k)$ contiennent une information juste mais aussi un certain "bruit". Intuitivement, on est conduit à penser que les données $f(x_k)$ contiennent
- une première composante variant lentement : c'est elle qui fournit l'information
- une deuxième composante variant très rapidement : c'est celle qui est due au "bruit" inévitable.

Tout l'art de l'ajustement des données consiste à éliminer la deuxième composante pour ne retenir que la "substantifique moëlle". Le principe de l'étude qui suit consiste à chercher la fonction f sous la forme

$$f(x) = c_1\theta_1(x) + c_2\theta_2(x) + ... + c_n\theta_n(x)$$

où les θ_i sont des fonctions connues (par exemple des polynômes) et les c_i des coefficients à déterminer. Implicitement, n sera assez grand pour contenir l'information, mais suffisamment petit pour se débarrasser des "bruits".

Ainsi dans l'exemple 3.6, on pourra déterminer c_1 et c_2 pour que les déviations

$$\{f(x_k) - (c_1 + c_2 x_k)\,; k = 0, ..., p\}$$

soient les plus petites possibles (contrairement à l'interpolation, il n'est pas possible d'annuler ces déviations). Maintenant, il existe plusieurs notions de "le plus petit possible". On pourrait chercher à minimiser :

$$\max_{0\leq k\leq p} |f(x_k) - (c_1 + c_2 x_k)|$$

ou encore

$$\sum_{0\leq k\leq p} |f(x_k) - (c_1 + c_2 x_k)|$$

(c'est-à-dire une norme $\|.\|\infty$ ou une norme $\|.\|_1$). Ceci conduira en général à des (c_1, c_2) différents et beaucoup plus difficiles à obtenir.

2.2.2 Méthode classique des moindres carrés

C'est une troisième voie qui est le plus souvent choisie dans la pratique, car elle conduit à un problème de minimisation quadratique en les inconnues c_i, à savoir minimiser

$$\sum_{0\leq k\leq p} |f(x_k) - (c_1 + c_2 x_k)|^2 \tag{2.16}$$

ou plus généralement minimiser

$$\sum_{0\leq k\leq p} \left(f(x_k) - (c_1\theta_1(x_k) + ... + c_n\theta_n(x_k))\right)^2. \tag{2.17}$$

Posons $E(c_1, c_2, ..., c_n) = \sum_{0\leq k\leq p} \left(f(x_k) - (c_1\theta_1(x_k) + ... + c_n\theta_n(x_k))\right)^2$. Si E admet un minimum en $c = (c_1, c_2, ..., c_n)$, on a

$$\frac{\partial E}{\partial c_i}(c) = 0, i = 1, ..., n$$

soit

$$\left\{ \begin{array}{l} 2\sum_{0\leq k\leq p} \theta_i(x_k) \left(f(x_k) - (c_1\theta_1(x_k) + ... + c_n\theta_n(x_k))\right) = 0 \\ i = 1, ..., n \end{array} \right\} \tag{2.18}$$

ce qui constitue un système linéaire de n équations aux n inconnues $c_1, c_2, ..., c_n$, appelées parfois équations normales. Donnons une autre interprétation, plus géométrique, de ce système (cf Figure 2.3). Notons

$$e = (e_0, e_1, ..., e_p), \quad e_k = f(x_k) - (c_1\theta_1(x_k) + ... + c_n\theta_n(x_k))$$

le vecteur erreur et $\theta_i = (\theta_i(x_0), \theta_i(x_1), ..., \theta_i(x_p))$. Alors, si on note $\langle u, v\rangle = \sum_{i=0}^{p} u_i v_i$ le produit scalaire dans $\mathbb{R}^{p+1}$, les équations (2.18) expriment

$$\forall i = 1, ..., n, \quad \langle e, \theta_i\rangle = 0 \tag{2.19}$$

en d'autre termes, le vecteur erreur e doit être orthogonal à tous les vecteurs θ_i, ce qui, géométriquement, revient à dire que le vecteur $c_1\theta_1 + c_2\theta_2 + ... + c_n\theta_n$ solution est la projection orthogonale du vecteur $P = (f(x_0), ..., f(x_p))$ sur le sous espace engendré par les θ_i.

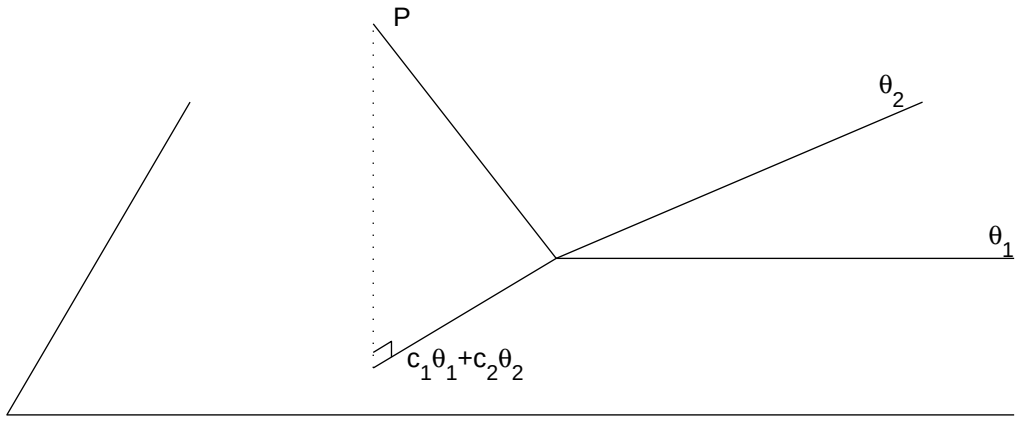

FIGURE 2.3 – Ajustement et projection orthogonale

Exemple 2.5 Ajustement linéaire : on prend $n = 2$ et $\theta_1(x) \equiv 1, \theta_2(x) \equiv x$. On cherche donc la fonction affine $c_1 + c_2 x$ la plus voisine au sens des moindres carrés des valeurs $(x_k, f(x_k))$.
Le système (2.18) devient

$$\begin{cases} \sum_{0\leq k\leq p} (f(x_k) - (c_1 + c_2 x_k)) = 0 \\ \sum_{0\leq k\leq p} x_k (f(x_k) - (c_1 + c_2 x_k)) = 0 \end{cases}$$

ou

$$\begin{cases} (p+1)c_1 + \left(\sum_{k=0}^p x_k\right) c_2 = \sum_{k=0}^p f(x_k) \\ \left(\sum_{k=0}^p x_k\right) c_1 + \left(\sum_{k=0}^p x_k^2\right) c_2 = \sum_{k=0}^p x_k f(x_k) \end{cases}$$

Notons :

$$m = \frac{\sum_{k=0}^p x_k}{p+1} \text{ (moyenne des } x_k), \quad m_2 := \frac{\sum_{k=0}^p x_k^2}{p+1} \text{ (moment d'ordre 2)}$$

$$\overline{f} := \frac{\sum_{k=0}^p f(x_k)}{p+1}, \quad \overline{\overline{f}} = \frac{\sum_{k=0}^p x_k f(x_k)}{p+1}.$$

Le système devient

$$\begin{cases} c_1 + m c_2 = \overline{f} \\ m c_1 + m_2 c_2 = \overline{\overline{f}}. \end{cases}$$

Son unique solution est donnée par :

$$c_1 = \frac{m_2 \overline{f} - m\overline{\overline{f}}}{m_2 - m^2}, \quad c_2 = \frac{\overline{\overline{f}} - m\overline{f}}{m_2 - m^2}.$$

Remarque 2.7 On vérifie que $m_2 \neq m^2$ si les x_k sont distincts. Dans le cas général, on montre aussi que le système (2.18) a toujours une solution. Il s'écrit

$$Ac = b \tag{2.20}$$

où $c = (c_1, ..., c_n)$, A est la matrice dont les coefficients sont $\{\langle \theta_j, \theta_i \rangle\}_{1 \leq i \leq n\ , 1 \leq j \leq n}$ (notons que c'est toujours une matrice symétrique) et $b = (b_1, ..., b_n)$ avec $b_i = \overline{\langle f, \theta_i \rangle}$.

Nous verrons dans le chapitre 6 les techniques de résolution de systèmes linéaires. Nous pouvons cependant déjà souligner que les méthodes sont d'autant plus performantes que la matrice est creuse (c'est-à-dire avec beaucoup de zéros) et à diagonale dominante (i.e. éléments hors diagonale petits devant ceux de la diagonale).
Un cas particulier très intéressant que nous allons développer ici est celui où on choisit les facteurs $\theta_i(x)$ de telle façon que les produits scalaires $\langle \theta_j, \theta_i \rangle$ soient nuls pour $i \neq j$, c'est-à-dire

$$\sum_{k=0}^{p} \theta_j(x_k)\theta_i(x_k) = 0, \quad \forall i \neq j. \tag{2.21}$$

Dans la pratique, on s'arrange pour être si possible dans cette situation (ou proche de cette situation). Sans précaution, on peut avoir des problèmes de conditionnement [2] très sérieux dans la résolution de (2.20). C'est le cas dans l'exemple suivant :

Exemple 2.6 Prenons $p = 3, \theta_1(x) \equiv 1, \theta_2(x) \equiv x, \theta_3(x) \equiv x^2$. On approche une fonction f sur l'intervalle $[10, 15]$ connaissant ses valeurs en 6 points dont les extrémités. On montre que le conditionnement associé (cf. section 7.1 au chapitre 7) peut être supérieur à 10^5.

Remarque 2.8 dans la formule (2.21), si les points x_k sont équidistants dans l'intervalle $[a, b]$, on a :

$$0 = \sum_{k=0}^{p} \theta_j \left(a + k\frac{b-a}{p} \right) \theta_i \left(a + k\frac{b-a}{p} \right) \sim \frac{p}{b-a} \int_a^b \theta_j(x)\theta_i(x)dx,$$

si p est assez grand. Ainsi, une version continue de notre problème consiste à considérer les polynômes (ou des fonctions plus générales) θ_i telles que

$$\int_a^b \theta_j(x)\theta_i(x)dx = 0.$$

Cette intégrale représente également un produit scalaire sur l'espace des fonctions et par analogie on dit que les deux fonctions sont orthogonales (le bon espace fonctionnel associé est alors $L^2([a, b])$).

Exemple 2.7 (a) les fonctions 1 et x sont orthogonales dans $L^2([-1, 1])$ pour le produit scalaire

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)dx$$

$$\text{car } \int_{-1}^{1} 1.xdx = \left[\frac{x^2}{2} \right]_{-1}^{1} = 0.$$

2. voir section 7.1.

(b) les fonctions $\cos kx$ et $\cos px$ sont orthogonales dans $L^2([0,2\pi])$ pour $|k| \neq |p|$ en effet :

$$\int_0^{2\pi} \cos kx \cos px dx = \frac{1}{2}\int_0^{2\pi} [\cos(k+p)x + \cos(k-p)x]\, dx = 0.$$

(c) les fonctions $\cos kx$ et $\sin px$ sont orthogonales dans $L^2([0,2\pi])$ en effet :

$$\int_0^{2\pi} \cos kx \sin px dx = \frac{1}{2}\int_0^{2\pi} [\sin(k+p)x + \sin(p-k)x]\, dx = 0.$$

2.2.3 Polynômes orthogonaux

Comme nous serons amenés à les utiliser dans le chapitre sur l'intégration numérique, nous allons maintenant étudier les suites de polynômes orthogonaux pour des produits scalaires du type :

$$\langle P, Q\rangle = \int_a^b P(x)Q(x)w(x)dx \tag{2.22}$$

où w est une fonction continue strictement positive sur $]a,b[$.
Si on suppose que $x^n w(x)$ est intégrable pour tout n, alors (2.22) définit un produit scalaire sur l'espace des polynômes.

Définition 2.1 *Par suite de polynômes orthogonaux, on entend une suite (finie ou infinie) $P_0(x), P_1(x), \ldots$ telle que*

$$P_i \text{ est de degré } i \tag{2.23}$$

$$\langle P_i, P_j\rangle = 0, \quad \text{si } i \neq j. \tag{2.24}$$

Proposition 2.1 *Soit (P_n) une suite de polynômes orthogonaux ; alors*
- *Tout polynôme Q de degré inférieur ou égal à k s'écrit de façon unique*

$$Q(x) = d_0P_0(x) + d_1P_1(x) + ... + d_kP_k(x) \qquad (\text{avec } d_i = \frac{\langle P_i, Q\rangle}{\langle P_i, P_i\rangle}). \tag{2.25}$$

- *Si Q est de degré $< k$, alors*

$$\langle Q, P_k\rangle = 0. \tag{2.26}$$

- *Si A_i désigne le coefficient du terme de plus haut degré de P_i, on a la relation de récurrence à trois termes*

$$\widehat{P}_{i+1}(x) = (x - B_i)\widehat{P}_i(x) - C_i\widehat{P}_{i-1}(x) \tag{2.27}$$

$$\text{où } \widehat{P}_i(x) := \frac{P_i(x)}{A_i} \quad \text{(polynôme normalisé)},$$

$$B_i = \frac{\left\langle x\widehat{P}_i(x), \widehat{P}_i(x)\right\rangle}{\left\langle \widehat{P}_i(x), \widehat{P}_i(x)\right\rangle},\ C_i = \frac{\left\langle \widehat{P}_i(x), \widehat{P}_i(x)\right\rangle}{\left\langle \widehat{P}_{i-1}(x), \widehat{P}_{i-1}(x)\right\rangle}.$$

- *P_i a exactement i zéros réels distincts.*

Démonstration : Le point (2.25) est classique en algèbre linéaire : les polynômes $P_0, ..., P_k$ forment une base de l'espace des polynômes de degré inférieur ou égal à k, car ils sont tous de degrés différents.
Pour (2.26), on remarque que, si $\deg Q < k$, d'après (2.25),

$$Q(x) = d_0 P_0(x) + d_1 P_1(x) + ... + d_{k-1} P_{k-1}(x)$$

et d'après la bilinéarité du produit scalaire

$$\langle Q, P_k \rangle = \sum_{i=0}^{k-1} d_i \langle P_i, P_k \rangle = 0.$$

Vérifions (2.27) : on peut toujours écrire

$$\widehat{P}_{i+1}(x) = x\widehat{P}_i(x) + \alpha_i \widehat{P}_i(x) + \alpha_{i-1}\widehat{P}_{i-1}(x) + ... + \alpha_0 \widehat{P}_0(x) \tag{2.28}$$

puisque $\left\{x\widehat{P}_i(x), \widehat{P}_i(x), \widehat{P}_{i-1}(x), ..., \widehat{P}_0(x)\right\}$, étant une famille de polynômes de degrés distincts, est une base de l'espace des polynômes de degré inférieur ou égal à $i+1$. D'autre part, puisque les polynômes sont normalisés, le coefficient de $x\widehat{P}_i(x)$ dans l'écriture $\widehat{P}_{i+1}$ ci-dessus est égal à 1.
Multiplions scalairement (2.28) par $\widehat{P}_i$; on obtient grâce à (2.24) :

$$0 = \left\langle x\widehat{P}_i(x), \widehat{P}_i(x) \right\rangle + \alpha_i \left\langle \widehat{P}_i(x), \widehat{P}_i(x) \right\rangle + 0$$

d'où

$$\alpha_i = -\frac{\left\langle x\widehat{P}_i(x), \widehat{P}_i(x) \right\rangle}{\left\langle \widehat{P}_i, \widehat{P}_i \right\rangle}.$$

Multiplions scalairement (2.28) par $\widehat{P}_j$ où $j \leq i-1$. Toujours grâce à (2.24), on a :

$$0 = \left\langle x\widehat{P}_i, \widehat{P}_j \right\rangle + \alpha_j \left\langle \widehat{P}_j, \widehat{P}_j \right\rangle.$$

Mais $\left\langle x\widehat{P}_i, \widehat{P}_j \right\rangle = \left\langle \widehat{P}_i, x\widehat{P}_j \right\rangle = 0$ si $j+2 \leq i$. Puisque $\left\langle \widehat{P}_j, \widehat{P}_j \right\rangle \neq 0$, ceci implique $\alpha_j = 0$ pour tout $j = 0, 1, ..., i-2$. Pour $j = i-1$, on remarque que

$$\left\langle x\widehat{P}_i, \widehat{P}_{i-1} \right\rangle = \left\langle \widehat{P}_i, x\widehat{P}_{i-1} \right\rangle = \left\langle \widehat{P}_i, \widehat{P}_i \right\rangle + \left\langle \widehat{P}_i, x\widehat{P}_{i-1} - \widehat{P}_i \right\rangle.$$

Mais ce dernier produit scalaire est nul puique $x\widehat{P}_{i-1} - \widehat{P}_i$ est un polynôme de degré inférieur ou égal à $i-1$. On en déduit

$$\alpha_{i-1} = -\frac{\left\langle \widehat{P}_i, \widehat{P}_i \right\rangle}{\left\langle \widehat{P}_{i-1}, \widehat{P}_{i-1} \right\rangle},$$

ce qui termine la démonstration de (2.27).

Pour le dernier point, notons $\zeta_1, ..., \zeta_k$ les racines réelles distinctes de multiplicité impaire et posons

$$R_i(x) = \begin{cases} 1 & \text{si } k = 0 \text{ (pas de racine réelle de multiplicité impaire)} \\ (x-\zeta_1)(x-\zeta_2)...(x-\zeta_k) & \text{si } k > 0. \end{cases}$$

Si $k = i$, P_i a alors i racines réelles distinctes ce qu'on voulait prouver.
Sinon $k < i$, et d'après (2.24)

$$\langle P_i, R_i \rangle = 0.$$

Pour le produit scalaire (2.22), ceci signifie

$$\int_a^b P_i(x) R_i(x) w(x) dx = 0.$$

Mais $P_i = R_i Q_i$ où Q_i est de signe constant d'après le choix de R_i ; ainsi

$$\left\{ \begin{array}{c} \int_a^b R_i(x)^2 Q_i(x) w(x) dx = 0 \\ R_i(x)^2 Q_i(x) w(x) \text{ de signe constant} \end{array} \right\} \Longrightarrow R_i^2 Q_i w \equiv 0;$$

ceci implique en particulier $Q_i = 0$, ce qui est contradictoire avec $\deg Q_i = i - k > 0$.

2.2.4 Exemples classiques de polynômes orthogonaux

(1) $[a, b] = [-1, 1]$ et $w(x) = (1-x)^\alpha (1+x)^\beta$ avec $\alpha > -1$ et $\beta > -1$: on obtient les polynômes de Jacobi. On leur attribue des noms différents dans les cas particuliers suivants :

- $\alpha = \beta = 0$: polynômes de Legendre. Une fois les polynômes normalisés, on a la relation de récurrence :
$$P_{n+1}(x) = \frac{(2n+1)xP_n(x) - nP_{n-1}(x)}{n+1}.$$
- $\alpha = \beta = -\frac{1}{2}$: on retrouve les polynômes de Chebyshev définis par la relation de récurrence
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$
- $\alpha = \beta = \frac{1}{2}$: polynômes de Chebyshev de seconde espèce (les précédents étant alors de première espèce).

(2) $[a, b[= [0, +\infty[$ et $w(x) = e^{-x}$: on a les polynômes de Laguerre. Plus généralement, si $w(x) = x^\alpha e^{-x}$ avec $\alpha > -1$, on a les polynômes de Laguerre généralisés. La relation de récurrence s'écrit :

$$P_{n+1}(x) = -\frac{1}{n+1}(x - 2n - \alpha - 1)P_n(x) + (n+\alpha)P_{n-1}(x).$$

(3) $]a, b[=]-\infty, +\infty[$, $w(x) = e^{-x^2}$: polynômes d'Hermite.

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x).$$

Remarque 2.9 on vérifie directement que les polynômes de Chebyshev sont orthogonaux pour le produit scalaire

$$\int_{-1}^{1} \frac{f(x)g(x)dx}{(1-x^2)^{\frac{1}{2}}}$$

car, en posant $x = \cos\theta$, on a :

$$\int_{-1}^{1} \frac{T_n(x)T_p(x)dx}{\sqrt{1-x^2}} = \int_0^{\pi} T_n(\cos\theta)T_p(\cos\theta)d\theta = \int_0^{\pi} \cos n\theta \cos p\theta d\theta = 0 \text{ si } n \neq p.$$

En ce qui concerne l'approximation au sens des moindres carrés d'une fonction f par des polynômes, on utilise généralement des polynômes orthogonaux plutôt que la base canonique usuelle $1, x, x^2...$ On a alors le résultat suivant :

Théorème 2.4 *Soit $f : [a,b] \to \mathbb{R}$ telle que $\langle f, f\rangle < \infty$ ou $\langle .\rangle$ est défini par (2.22). Alors, pour tout $k \geq 1$ (resp. $k \leq N-1$), si $P_0, P_1, ..., P_k$ est une suite de polynômes orthogonaux pour le produit scalaire $\langle .\rangle$, il existe un et un seul polynôme Q de la forme*

$$Q(x) = c_0P_0(x) + c_1P_1(x) + ... + c_kP_k(x)$$

minimisant $\langle f-Q, f-Q\rangle$. Les coefficients sont donnés par

$$c_i = \frac{\langle P_i, f\rangle}{\langle P_i, P_i\rangle}, \quad i = 0, ..., k.$$

Remarque 2.10 $\langle f-Q, f-Q\rangle = \|f-Q\|^2$ représente le carré de la distance de f à Q pour la norme hilbertienne associée au produit scalaire $\langle .\rangle$.

Exemple 2.8 Trouver le polynôme P de degré inférieur ou égal à 3 qui minimise

$$\int_{-1}^{1} (e^x - P(x))^2 \, dx.$$

On utilise comme base de polynômes les polynômes de Legendre, soit

$$L_0(x) = 1, \; L_1(x) = x, \; L_2(x) = \frac{3}{2}\left(x^2 - \frac{1}{3}\right), \; L_3(x) = \frac{5}{2}\left(x^3 - \frac{3}{5}x\right).$$

On calcule alors

$$\begin{aligned}
\langle f, L_0\rangle &= \int_{-1}^{1} e^x dx = e - \frac{1}{e} \\
\langle f, L_1\rangle &= \int_{-1}^{1} xe^x dx = \frac{2}{e} \\
\langle f, L_2\rangle &= \frac{3}{2}\int_{-1}^{1} e^x \left(x^2 - \frac{1}{3}\right) dx = e - \frac{7}{e} \\
\langle f, L_3\rangle &= \frac{5}{2}\int_{-1}^{1} e^x \left(x^3 - \frac{3}{5}\right) dx = -5e + \frac{37}{e}.
\end{aligned}$$

D'autre part, on montre que $\langle L_i, L_i\rangle = \frac{2}{2i+1}$. On en déduit le polynôme solution :

$$P(x) = 1,175201194L_0(x) + 1,10363824L_1(x) + 0,3578143506L_2(x) + 0,07045563367L_3(x).$$

Sous forme canonique usuelle, on obtient :

$$P(x) = 0,9962940183 + 0,9979548730x + 0,5367215260x^2 + 0,1761390842x^3.$$

Remarque 2.11 un avantage de l'écriture de P sur la base L_0, L_1, L_2, L_3 est que par exemple la partie tronquée $1,175...L_0 + 1,131...L_1$ est la meilleure approximation linéaire au sens des moindres carrés de e^x sur $[-1, 1]$ (un peu comme dans le cas de la formule d'interpolation de Newton).

2.2.5 Polynômes trigonométriques

Théorème 2.5 (Décomposition en série de Fourier) *Soit $f : [-\pi, \pi] \to \mathbb{R}$ continue. Alors il existe une unique meilleure approximation au sens des moindres carrés dans l'espace des polynômes trigonométriques de la forme*

$$S(x) = a_0 + \sum_{k=1}^{n} a_k \cos kx + \sum_{k=1}^{n} b_k \sin kx. \tag{2.29}$$

Ses coefficients sont donnés par :

$$\begin{cases} a_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)dx \\ a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx dx \\ b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx dx \end{cases} \tag{2.30}$$

Remarque 2.12 On reconnait bien sûr là les classiques coefficients de Fourier !

Démonstration : Il s'agit de minimiser $\int_{-\pi}^{\pi} (f(x) - S(x))^2 \, dx$. Comme les fonctions $\{1, \cos x, \cos 2x, \ldots, \cos nx, \sin x, sin2x, \ldots, \sin nx\}$ forment une suite orthogonale par rapport au produit scalaire

$$\langle f, g \rangle = \int_{-\pi}^{\pi} f(x)g(x)dx,$$

d'après l'analyse faite en début de chapitre, on a immédiatement une solution unique donnée par :

$$\begin{aligned} a_0 &= \frac{\langle f, 1 \rangle}{\langle 1, 1 \rangle} = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)dx \\ a_k &= \frac{\langle f, \cos kx \rangle}{\langle \cos kx, \cos kx \rangle} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx dx \\ b_k &= \frac{\langle f, \sin kx \rangle}{\langle \sin kx, \sin kx \rangle} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx dx. \end{aligned}$$

Remarque 2.13 Ce type d'approximation est surtout utilisé pour les fonctions périodiques comme celles représentant les divers phénomènes ondulatoires. En général, une fonction est composée d'un très grand nombre de fréquences : en se limitant à un nombre fini n, on "filtre" l'information en ne gardant que les plus basses fréquences ("information") et en se débarrassant du "bruit" constitué surtout par les hautes fréquences.

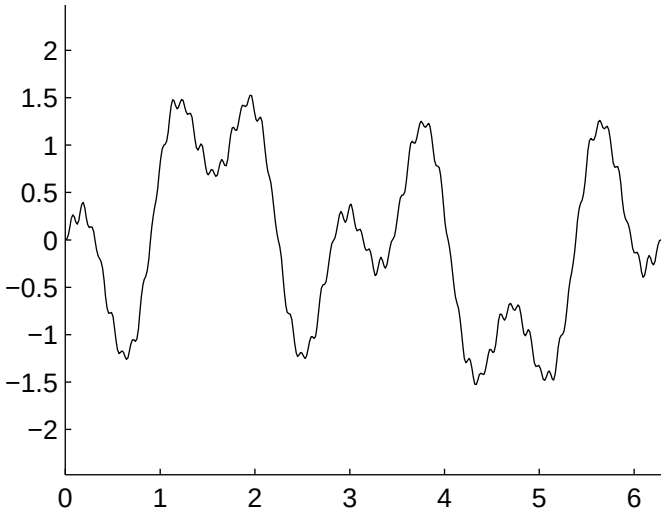

FIGURE 2.4 – Signal bruité

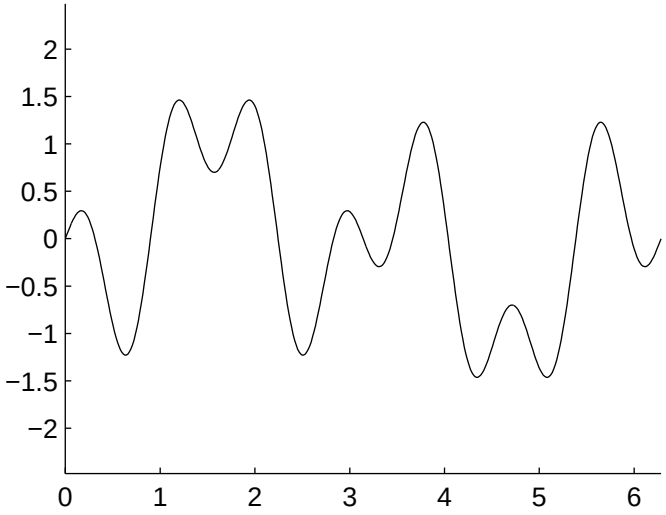

FIGURE 2.5 – Signal débruité

ainsi

devient

Remarque 2.14 Si une fonction f est de période T, on se ramène à l'intervalle $[-\pi, \pi]$ en remarquant que $\widetilde{f}(x) = f\left(\frac{T}{2\pi}x\right)$ est périodique de période 2π. Elle est donc connue pour tout x si elle est connue pour $x \in [-\pi, \pi]$.

On utilise souvent la représentation complexe des coefficients de Fourier :

$$f_n(x) = \sum_{j=-n}^{n} \widehat{f}(j) e^{ijx} \tag{2.31}$$

où

$$\widehat{f}(j) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ijx} dx \qquad j = -n, ..., n$$

est une autre expression du polynôme trigonométrique qui approche le mieux f dans $L^2([0,\pi])$. On a d'ailleurs

$$f(x) = \lim_{n\to\infty} f_n(x) = \sum_{j=-\infty}^{+\infty} \widehat{f}(j) e^{ijx} \tag{2.32}$$

la limite ci-dessus étant comprise au sens de la norme

$$\|f - f_n\|_2 = \sqrt{\int_{-\pi}^{\pi} (f(x) - f_n(x))^2 \, dx}$$

pour toute fonction f dans $L^2([0,\pi])$.

De nombreux calculs nécessitent la connaissance des coefficients de Fourier de f. Notons d'abord, que si f est 2π-périodique, ils s'écrivent encore

$$\widehat{f}(j) = \frac{1}{2\pi} \int_{0}^{2\pi} f(x) e^{-ijx} dx.$$

Alors une formule de quadrature très élémentaire conduit à prendre comme approximation

$$\widehat{f}_N(j) = \frac{1}{2\pi} \frac{2\pi}{N} \sum_{n=0}^{N-1} f\left(\frac{2\pi n}{N}\right) e^{-ij(2\pi n/N)}.$$

On montre que cette approximation est convenable si

$$\frac{N}{2} \geq |j| \, .$$

Cette condition se comprend par exemple si $f(x) \equiv 1$; il s'agit alors d'intégrer la fonction $x \to e^{-ijx}$ qui est $\frac{2\pi}{j}$-périodique. Il est clair que les points d'évaluation de la fonction doivent avoir une fréquence plus rapide que celle de la fonction elle-même pour être significatifs.
Nous allons maintenant nous concentrer sur le calcul numérique d'expressions du type

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} f(x_k) e^{-ijx_k}, \qquad x_k = \frac{2\pi k}{N}, \qquad |j| \leq \frac{N}{2}. \tag{2.33}$$

Fait sous la forme ci-dessus, le calcul requiert l'évaluation des produits $f(x_k)e^{-ijx_k}$ puis N sommes et une division pour obtenir c_j. Ceci doit être effectué pour chaque c_j donc

environ N fois. Nous négligerons les divisions et nous compterons les opérations élémentaires consistant à calculer un produit $f(x_k)e^{-ijx_k}$ et à l'ajouter à une somme partielle précédente. Ce nombre d'opérations élémentaires est donc ici de l'ordre de N^2. Ainsi un choix de 1000 points nécessite un million d'opérations. Ceci a été un obstacle majeur à l'utilisation de l'analyse de Fourier jusqu'à ce que Cooley et Tukey proposent une méthode qui, après quelques améliorations, a fait passer le nombre d'opérations élémentaires à un ordre $O(N \log N)$ (soit autour de quelques milliers si $N = 1000$) : il s'agit de la Transformée de Fourier rapide (F.F.T.=Fast Fourier Transform) que nous expliquons maintenant

2.2.6 La transformée de Fourier rapide

Nous en décrirons seulement le principe qui repose sur une décomposition de l'entier N : nous supposerons ici que $N = PQ$, P et Q entiers supérieurs ou égaux à 2. Notons

$$w_N = e^{-i2\pi/N}, \quad w_q = e^{-i2\pi/Q}, \quad Z = [Z_1, ..., Z_N]$$

Avec des changements de notation évidents à partir de (2.33), il s'agit de calculer le vecteur "Transformée de Fourier discrète"

$$\widehat{Z} = \left[\widehat{Z}_1, ..., \widehat{Z}_N\right]$$

avec

$$\widehat{Z}_j := \sum_{k=1}^{n} Z_k w_N^{(j-1)(k-1)} \qquad j = 1, ..., N$$

On interprète le tableau unidimensionnel Z comme un tableau bidimensionnel $Z = Z(P, Q)$ avec :

$$\begin{aligned} 1 &\leq p \leq P \\ 1 &\leq q \leq Q \\ Z(p,q) &= Z_{p+P(q-1)}. \end{aligned}$$

Alors

$$\widehat{Z}_j = \sum_{p,q} Z(p,q) w_N^{(j-1)(p-1+P(q-1))} = \sum_{p=1}^{P} \left[\sum_{q=1}^{Q} Z(p,q) w_Q^{(j-1)(q-1)} \right] w_N^{(j-1)(p-1)}$$

puisque $w_N^P = e^{-i2\pi P/N} = e^{-i2\pi/Q} = w_Q$. Il se trouve et c'est le point important, que la somme entre crochets est une fonction de j périodique de période Q car

$$w_Q^{(j+Q-1)(q-1)} = w_Q^{j(q-1)} w_Q^{Q(q-1)} = w_Q^{j(q-1)}$$

puique $w_Q^Q = 1$. Ainsi seul intervient le reste $\widehat{j}$ de la division de j par Q :
$j = \lambda Q + \widehat{j}$ $0 \leq \widehat{j} \leq Q$, et il suffit de calculer la somme entre crochets pour $j = \widehat{j}$ variant de 1 à Q.
Ce calcul nécessite Q opérations pour chaque j et p et doit donc être effectué $Q \times P$ fois d'où $Q^2 P = NQ$ opérations pour le calcul des crochets. Ensuite le calcul des $\widehat{Z}_j$ nécessite P

opérations et doit être effectué N fois soit NP opérations. L'ensemble des $\widehat{Z}_j$, $j = 1, ..., N$ est donc obtenu avec $N(P+Q)$ opérations ce qui est déjà bien moins que N^2 opérations.

	$N = 1\ 000$	$N^2 = 1\ 000\ 000$
Exemple :	$N = 10 \times 100 = P \times Q$	$N(P+Q) = 1\ 000 \times 110 = 110\ 000$
	$N = 25 \times 40 = P \times Q$	$N(P+Q) = 1\ 000 \times 65 = 65\ 000$

On peut encore diminuer le nombre d'opérations en décomposant N sous la forme $N = P_1 \times P_2 \times ... \times P_k$ est en répétant l'idée ci-dessus. Un cas fréquemment utilisé est celui où $N = 2^k$. Nous renvoyons à la littérature pour les détails de la méthode.

2.3 Approximation par des fonctions polynômiales par morceaux

Etant donnée $f : [a, b] \to \mathbb{R}$ connue en $(n+1)$ points $a = x_0 < x_1 < ... < x_n = b$, on l'approche par des fonctions continues dont la restriction à chaque intervalle $[x_{i-1}, x_i]$ est polynômiale.

2.3.1 Approximation linéaire par morceaux

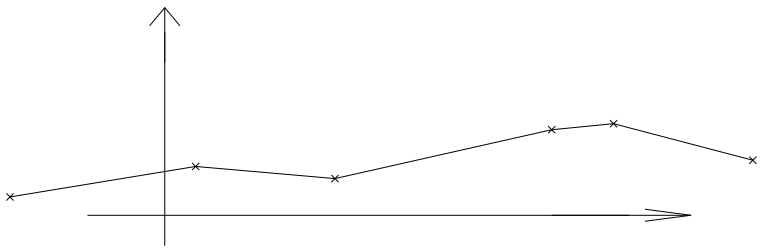

FIGURE 2.6 – Approximation linéaire par morceaux

La restriction de l'approximation P à l'intervalle $[x_{i-1}, x_i]$ est donnée par :

$$P_i(x) = f[x_{i-1}] + f[x_{i-1}, x_i](x - x_{i-1}).$$

La fonction P ainsi construite est évidemment continue sur l'intervalle $[a, b]$. Du temps des tables de fonctions élémentaires (tables de logarithmes ou de fonctions trigonométriques), on obtenait les valeurs hors table par interpolation affine, ce qui revenait exactement à remplacer la fonction par son approximation linéaire par morceaux.

2.3.2 Approximation cubique

Les polynômes de base sont de degré 3. On peut supposer que f est connue en les points $[x_{i-1}, x_{i-2/3}, x_{i-1/3}, x_i]$ où $x_{i-1/3}, x_{i-2/3}$ sont des points distincts de $]x_{i-1}, x_i[$. On prend alors pour P_i l'interpolé de f sur $[x_{i-1}, x_i]$ en ces 4 points.

Une situation plus intéressante consiste à considérer que, outre les valeurs $f(x_i)$, $i = 0, 1, ...N$, on connait aussi les valeurs $f'(x_i)$, $i = 0, 1, ...N$. Dans ce cas, on pourra prendre pour P_i le polynôme de degré 3 tel que :

$$\begin{cases} P_i(x_{i-1}) = f(x_{i-1}) & P_i(x_i) = f(x_i) \\ P_i'(x_{i-1}) = f'(x_{i-1}) & P_i'(x_i) = f'(x_i) \end{cases} \tag{2.34}$$

on sort alors du domaine de l'interpolation de Lagrange pour entrer dans ce qu'on appelle l'interpolation d'Hermite.

Théorème 2.6 *Il existe un et un seul polynôme de degré* 3 *satisfaisant (2.34). Il est donné par la formule de Newton*

$$\begin{cases} P_i(x) = f[x_{i-1}] + f[x_{i-1}, x_{i-1}](x - x_{i-1}) + f[x_{i-1}, x_{i-1}, x_i](x - x_{i-1})^2 \\ \qquad + f[x_{i-1}, x_{i-1}, x_i, x_i](x - x_{i-1})^2(x - x_i) \end{cases} \tag{2.35}$$

où on définit

$$f[x_{i-1}, x_{i-1}] := \lim_{h\to 0} f[x_{i-1}, x_{i-1} + h] = \lim_{h\to 0} \frac{f[x_{i-1} + h] - f[x_{i-1}]}{h} = f'(x_{i-1})$$

les autres différences divisées étant définies de la façon habituelle.

Remarque 2.15 Ce résultat se généralise bien sûr à un nombre de points quelconque avec des dérivées d'ordre quelconque. La formule de Newton est la même si on étend la définition des différences divisées par continuité comme ci-dessus, soit par récurrence

$$\begin{cases} f[x_i] := f(x_i) \\ f[x_0, x_1, ..., x_k] := \begin{cases} \text{formule usuelle si } x_k \neq x_0 \\ \lim_{h\to 0} f[x_0, ..., x_{k-1}, x_k + h] \text{ si } x_0 = x_k. \end{cases} \end{cases} \tag{2.36}$$

Nous renvoyons à la littérature pour la démonstration du Théorème 2.6 et sa version plus générale évoquée dans la remarque ci-dessus. Comme souvent dans la pratique, on ne dispose pas des valeurs de la dérivée, on peut dans la formule (2.34) remplacer $f'(x_i)$ par une valeur approchée s_i de $f'(x_i)$ par exemple

$$s_i = \frac{h_i f[x_i, x_{i+1}] + h_{i+1} f[x_{i-1}, x_i]}{h_i + h_{i+1}} \tag{2.37}$$

avec $h_i = x_i - x_{i-1}$. On parle alors d'interpolation cubique par morceaux de Bessel.
Notons que dans ce cas, la fonction d'approximation obtenue est non seulement continue, mais à dérivée continue puisque :

$$P_i'(x_i) = P_{i+1}'(x_i) = s_i.$$

2.3.3 Fonctions splines

Il se trouve que, au lieu de s'imposer les s_i sous la forme (2.37), on peut les choisir pour que la fonction d'approximation soit non seulement à dérivée continue, mais à dérivée seconde continue : il s'agit alors de fonctions splines d'interpolation. Le mot "spline" est

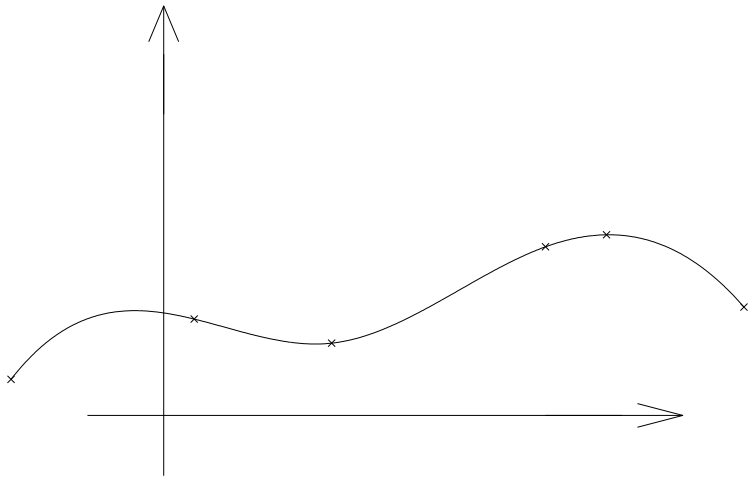

FIGURE 2.7 – Spline cubique

utilisé en anglais pour désigner la réglette flexible que les dessinateurs utilisent pour tracer une courbe d'allure "lisse" passant par des points donnés. La méthode que nous allons décrire correspond d'ailleurs à un "lissage" de la courbe passant par les points donnés de coordonnées $(x_i, f(x_i))$. Afin d'analyser cette méthode réécrivons le polynôme $P_i(x)$ tel que

$$\begin{cases} P_i(x_{i-1}) = f(x_{i-1}) & P_i(x_i) = f(x_i) \\ P_i'(x_{i-1}) = s_{i-1} & P_i'(x_i) = s_i. \end{cases}$$

Il est donné par la formule (2.35) où on remplace $f'(x_i)$ (resp ; $f'(x_{i-1})$) par s_i (resp. s_{i-1}), soit

$$\begin{aligned} P_i(x) &= f(x_{i-1}) + s_{i-1}(x - x_{i-1}) + h_i^{-1}\left(f[x_{i-1}, x_i] - s_{i-1}\right)(x - x_{i-1})^2 \\ &\quad + C_i (x - x_{i-1})^2 (x - x_i) \end{aligned}$$

avec

$$\begin{aligned} C_i &= \frac{f[x_{i-1}, x_i, x_i] - f[x_{i-1}, x_{i-1}, x_i]}{h_i} \\ &= \left(\frac{s_i - f[x_{i-1}, x_i]}{h_i} - \frac{f[x_{i-1}, x_i] - s_{i-1}}{h_i}\right)/h_i \\ &= \frac{s_i + s_{i-1} - 2f[x_{i-1}, x_i]}{h_i^2}. \end{aligned}$$

On cherche à réaliser

$$P_i''(x_i) = P_{i+1}''(x_i). \tag{2.38}$$

Puisque $\left((x - x_{i-1})^2 (x - x_i)\right)'' = 2(x - x_i) + 4(x - x_{i-1})$, la condition (2.38) s'écrit :

$$2\frac{f[x_{i-1}, x_i] - s_{i-1}}{h_i} + 4\frac{s_i + s_{i-1} - 2f[x_{i-1}, x_i]}{h_i} = 2\frac{f[x_i, x_{i+1}] - s_i}{h_{i+1}} - 2\frac{s_{i+1} + s_i - 2f[x_i, x_{i+1}]}{h_{i+1}}.$$

Ceci conduit au système linéaire en les s_i :

$$\begin{cases} h_{i+1}s_{i-1} + 2(h_i + h_{i+1})s_i + h_i s_{i+1} = 3(h_{i+1} f[x_{i+1}, x_i] + h_i f[x_i, x_{i+1}]) \\ i = 1, ..., N-1 \end{cases}$$

Ce qui fait $N-1$ équations aux inconnues $s_0, s_1, ..., s_N$. On peut se fixer arbitrairement les valeurs extrêmes s_0 et s_N (par exemple en écrivant que $P''(a) = P''(b) = 0$). On obtient alors un système linéaire à diagonale strictement dominante (|coeff de s_i| > |coeff de s_{i-1}|+ |coeff de s_{i-1}|, voir chapitre 6) ; donc il admet une solution unique. Ceci prouve l'existence d'une fonction spline cubique à dérivée seconde continue comme prévu.

Calcul d'erreur : On peut montrer que

$$\max_{a \le x \le b} |f(x) - P(x)| \le \max_{a \le \zeta \le b} \left| f^{(4)}(\zeta) \right| \frac{5 (\max_i h_i)^4}{384}$$

ce qui est une très bonne approximation si la dérivée $f^{(4)}$ est raisonnable et si les intervalles sont petits. D'autre part, au facteur 5 près, cette erreur est du même ordre que celle obtenue dans l'interpolation d'Hermite (selon (2.35)) alors que dans ce cas on utilise deux fois plus d'information sur la fonction, car on a besoin de $f(x_i)$ et $f'(x_i)$ pour tout i.
Ceci suggère que les fonctions splines doivent avoir aux points x_i des dérivées voisines de celles de f. On montre en effet que

$$\max_{a \le x \le b} \left| f'(x) - P'(x) \right| \le \max_{a \le \zeta \le b} \left| f^{(4)}(\zeta) \right| \frac{(\max_i h_i)^3}{24}.$$

Ceci fournit donc une méthode de calcul numérique de la dérivée d'une fonction en des points x_i tout à fait intéressante.

2.4 Exercices du chapitre 2

Exercice 2.1 On veut éditer une table de cosinus pour laquelle l'utilisateur cherchera la valeur d'un cosinus donné par interpolation linéaire entre les valeurs tabulées. Sachant qu'on veut obtenir des valeurs avec une erreur inférieure à 10^{-6}, combien d'entrées dans la table sont-elles nécessaires ?

Exercice 2.2 Déterminer la meilleure approximation linéaire sur $[0, 1]$ de $f(x) = x^5 + x^3$ au sens des moindres carrés.

Exercice 2.3 Le tableau suivant donne les valeurs d'une fonction f en trois points.

x	$f(x)$
$x_1 = \sqrt{17} - 4$	0.3771784
$x_2 = \sqrt{10} - 3$	0.4880117
$x_3 = \sqrt{5} - 2$	0.6754903

a) Déterminer une valeur approchée de f en $x_0 = \frac{x_2 + x_3}{2}$, par interpolation en x_2 et x_3, puis par interpolation en x_1, x_2 et x_3.
b) Donner dans les deux cas un majorant de l'erreur en fonction des dérivées de f.
c) On constate que les deux fonctions $f_1(x) = \sin \pi x$ et $f_2(x) = \sin \frac{\pi}{x}$ vérifient toutes deux le tableau de valeurs ci-dessus. Calculer $f_1(x_0)$ et $f_2(x_0)$. Expliquer et vérifier la cohérence avec les résultats numériques de a) et b).

Exercice 2.4 On donne les points $(0,2); (1,1); (2,\frac{2}{3}); (3,\frac{1}{2}); (4,\frac{1}{3}); (10,0.1)$.
Chercher des fonctions approchant au mieux ces points au sens des moindres carrés. Comparer les erreurs.

Exercice 2.5 Le tableau suivant donne la viscosité μ (mesurée) de l'éthylène en fonction de la température (en degrés Farenheit) :

T	0	50	100	150	200
μ	242	82.1	30.5	12.6	5.57

Proposer une loi décrivant μ en fonction de T.

2.5 TD2 : Interpolation et moindres carrés

Exercice 1
a) Soit $X = [x_1, x_2, \cdots, x_n]$ un vecteur contenant n points d'interpolation et
$Y = [f(x_1), f(x_2), \cdots, f(x_n)]$ un vecteur contenant les valeurs d'une fonction f en ces points. On veut calculer le polynôme d'interpolation de f en les points x_i à l'aide de la formule de Newton qu'on écrit ici aux points $x_n, x_{n-1}, ..., x_1$ soit

$$P_n(x) = \sum_{i=1}^{n} f[x_i, ..., x_n] \prod_{k=i+1}^{n} (x - x_k).$$

Écrire une fonction Matlab qui calcule $P_n(x)$ en utilisant l'algorithme suivant

Algorithme 1 : on suppose les valeurs $f(x_i)$ mises en mémoire dans d_i.

on obtient ici petit à petit les différences divisées $d_i = f[x_i, ..., x_n]$ ⟵

```
⌈ ⌈   de k = 1 à n − 1
  ⌈   de i = 1 à n − k
  ⌊   d_i := (d_{i+1} − d_i)/(x_{i+k} − x_i)
  ⌈   p := d_1
  ⌈   de i = 2 à n
  ⌊   p := d_i + (x − x_i) × p
⌊ P_n(x) := p.
```

algorithme de Hörner pour évaluer le polynôme au point x ⟵

b) Ecrire deux fonctions qui, étant donné un intervalle $[a, b]$, et un entier n génèrent respectivement :
- Les $n+1$ points équidistants : $x_i = a + \frac{b-a}{n}(i-1)$ avec $1 \leq i \leq n+1$.
- Les $n+1$ points de Chebyshev : $x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\frac{(2i-1)\pi}{2n+2}$ avec $1 \leq i \leq n+1$.

c) Soit f la fonction définie sur $[-5, 5]$ par :

$$f(x) = \frac{1}{1+x^2}$$

Tracer sur un même graphique les graphes de f, du polynôme d'interpolation associé à $n+1$ points équidistants et du polynôme d'interpolation associé à $n+1$ points de Chebyshev. Que constate-t-on lorsque n varie ?

Exercice 2
Le tableau suivant donne les résultats de mesure de la viscosité μ de l'éthylène pour différentes températures (exprimées en degrés Farenheit) :

T	0	50	100	150	200
μ	242	82.1	30.5	12.6	5.57

1. On souhaite modéliser la dépendance de la viscosité en fonction de la température par une loi du type : $\mu = A\exp(\alpha T)$.
Remarquer que $\nu = \log(\mu)$ suit alors une loi linéaire et en déduire les valeurs A_1 et α_1

qui minimisent l'erreur : $E_\nu = \Sigma|\log(\mu_i) - (\alpha T_i + \log(A))|^2$. On utilisera directement la fonction Matlab d'ajustement *polyfit*.

2. A partir de ces valeurs, déterminez les valeurs A_2 et α_2 qui minimisent l'erreur : $E_\mu = \Sigma|\mu_i - A\exp(\alpha T_i)|^2$. On utilisera pour cela la fonction Matlab de minimisation *fminsearch*.

2.6 Aide-mémoire

2.6.1 Polynômes

Un polynôme de degré n, n étant un entier positif, est une expression (bien souvent considérée comme une fonction) de la forme

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

où les coefficients $a_0, a_1, \ldots, a_n$ sont (dans ce cours) des nombres réels (ou plus rarement des nombres complexes). L'ensemble des polynômes de degré inférieur ou égal à n est un espace vectoriel pour la somme usuelle et le produit par un scalaire. On le notera $\mathbb{R}_n[x]$ Il est de dimension $n+1$ et a pour base *canonique* la famille $1, x, x^2, \ldots, x^n$.
Il existe bien d'autres bases qu'on peut utiliser dans d'autres circonstances. Ainsi, toute famille de polynômes $P_0, P_1, \ldots, P_n$ **de degré échelonné**, c'est-à-dire que le degré de P_k est exactement k, forme une base de $\mathbb{R}_n[x]$.

Un polynôme de degré n a au plus n racines réelles et il a exactement n racines complexes. Si son degré est pair, il peut n'avoir aucune racine réelle (c'est le cas par exemple de $(x^2+1)^n$), tandis que si son degré est impair, il a au moins une racine réelle, d'après le théorème des valeurs intermédiaires (si $a_n > 0$, $P(x)$ tend vers $+\infty$ quand $x \to +\infty$ et $P(x)$ tend vers $-\infty$ quand $x \to -\infty$ donc P doit s'annuler sur $\mathbb{R}$).

2.6.2 Théorème des accroissements finis

Dans ce chapitre, il est fait allusion au Théorème de Rolle ou au théorème des accroissements finis. Nous rappelons leur énoncé.
Théorème de Rolle : si f est continue sur $[a,b]$ et dérivable sur $]a,b[$ avec $f(a) = f(b)$, alors il existe $c \in]a,b[$ tel que $f'(c) = 0$.

Théorème des accroissements finis : si f est continue sur $[a,b]$ et dérivable sur $]a,b[$, alors il existe $c \in]a,b[$ tel que $f'(c) = (f(b) - f(a))/(b-a)$.

Notez que ces deux théorèmes sont clairement équivalents et que la formule de Taylor-Lagrange est une conséquence du théorème des accroissement finis.

2.6.3 Norme

Une norme dans un espace vectoriel E est une application $N : E \to \mathbb{R}_+$ qui vérifie

1. $N(x) = 0 \Rightarrow x = 0$
2. $\forall \lambda \in \mathbb{R}, \forall x \in E,\ N(\lambda x) = |\lambda|\, N(x)$
3. $\forall x, y \in E,\ N(x+y) \leq N(x) + N(y)$ (inégalité triangulaire).

On note souvent $\|x\|$ au lieu de $N(x)$.

Sur l'espace usuel $E = \mathbb{R}^N$, les normes les plus utilisées sont, pour $X = (x_1, x_2, \ldots, x_N)$
– $\|X\|_1 = \sum_{k=1}^{N} |x_k|$,
– $\|X\|_\infty = \max\{|x_k|\}$,

– et la norme euclidienne : $\|X\|_2 = \sqrt{\sum_{k=1}^{N} |x_k|^2}$.

La norme euclidienne dérive en fait du produit scalaire : $(X, Y) = \sum_{k=1}^{N} x_k y_k$ et fait de $\mathbb{R}^N$ un espace euclidien où on définit la notion d'orthogonalité, de projection orthogonale...

Chapitre 3

Dérivation et intégration numérique

3.1 Introduction

Le but de ce chapitre est double :
- indiquer comment calculer de façon approchée une dérivée, avec application à la résolution d'équations différentielles et d'équations aux dérivées partielles par la **méthode des différences finies**
- décrire les méthodes de base pour le calcul numérique d'intégrales (rectangles, trapèzes, Simpson, Gauss).

Un point commun de ces deux thèmes qui vont nous occuper est l'utilisation de polynômes d'interpolation. L'idée sous-jacente est simple : la fonction avec laquelle on travaille est trop compliquée ou pas assez connue pour qu'on puisse faire des opérations simples comme la dérivation ou l'intégration, on choisit alors de la remplacer par un polynôme et on dérive ou on intègre celui-ci. On commencera par présenter les formules de dérivation approchée qui en résultent avec un aperçu sur les méthodes de différences finies. On envisagera ensuite le calcul d'intégrales du type

$$\int_a^b f(x)\, w(x)\, dx \tag{3.1}$$

où w est une fonction poids (continue strictement positive sur $]a,b[$) et $f(x)\, w(x)$ est intégrable sur $[a,b]$.

Les méthodes d'intégration numérique que nous décrirons consistent toutes à remplacer l'intégrale (3.1) par une expression le plus souvent de la forme :

$$\sum_{i=0}^{N} A_i f(x_i), \tag{3.2}$$

où les x_i sont des points distincts de $[a,b]$ et A_i des coefficients réels, le tout choisi pour que la différence

$$E = \int_a^b f(x)\, w(x)\, dx - \sum_{i=0}^{N} A_i f(x_i)$$

soit petite.

Dans la pratique, il faut évidemment faire la différence entre le cas où la fonction f est connue en un nombre fini de points y_0, $y_1, ..., y_p$, auquel cas les points x_i devront être des points y_j et celui où la fonction f est connue analytiquement auquel cas on pourra choisir au mieux les points x_i (exemple : calcul de $\int_0^1 e^{-x^2} dx$).

Deux types d'approches sont à retenir :

- Méthodes composites (ou composées) : on divise l'intervalle $[a, b]$ en sous intervalles $[x_{i-1}, x_i]$ avec $a = x_0 < x_1 < ... < x_N = b$. Sur chaque intervalle $[x_{i-1}, x_i]$ on "remplace" $f(x)$ par un polynôme d'interpolation $P_{i,k}$ de degré $\leq k$ et on remplace $\int_{x_{i-1}}^{x_i} f(x)\,dx$ par $\int_{x_{i-1}}^{x_i} P_{i,k}(x)\,dx$ (ici $w(x) \equiv 1$) : ainsi, sur chaque intervalle $[x_{i-1}, x_i]$, on applique une méthode d'intégration élémentaire.
- Méthode de Gauss : elles s'appliquent pour des intervalles $[a, b]$ et des poids w particuliers : on approche f sur $[a, b]$ par un polynôme d'interpolation aux points de $[a, b]$ obtenus à partir des zéros de polynômes orthogonaux associés au poids w sur $[a, b]$. Cette méthode peut être appliquée directement sur l'intervalle $[a, b]$, ou, si $b - a$ est grand, sur des sous-intervalles d'une décomposition.

3.2 Dérivation numérique

Dans ce paragraphe, la fonction f n'est bien sûr pas connue par une formule explicite mais ou bien

- par ses valeurs sur un ensemble discret (en supposant les points assez proches pour que la notion de dérivée ait un sens). Le cas où la fonction f que l'on recherche est solution d'une certaine équation différentielle (ou aux dérivées partielles) rentre dans cette situation, car alors on cherche à estimer les valeurs de f en un nombre fini de points.
- par un algorithme de calcul ou une formule compliquée qui permet, au moins en théorie, de la calculer en tout point. On suppose bien sûr que la dérivée n'est pas accessible par un procédé analogue. A ce sujet, existent maintenant des techniques de différentiation automatique (par exemple le logiciel Odyssée développé par l'INRIA) qui permettent de différentier chaque pas d'un programme et ainsi de calculer une notion de dérivée, même pour une fonction définie par un programme. Nous ne traiterons pas cette notion ici, nous contentant d'une vision plus ancienne et classique.

Dans toute la suite, on supposera f connue ou calculable aux points $\ldots, x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, \ldots$ qu'on supposera proches. On notera $h_i = x_{i+1} - x_i$.

3.2.1 Dérivée première

Supposons qu'on veuille calculer une valeur approchée de $f'(x_i)$. Une première idée, déjà évoquée dans le chapitre précédent, consiste à remplacer f par une fonction spline passant par les mêmes points et à prendre la dérivée de la fonction spline. D'un point de vue numérique, cette idée est très bonne, assez stable et donne de bons résultats en terme d'erreur. Malheureusement, elle est assez lourde à mettre en place, nécessite beaucoup de calculs. Elle a de plus un petit côté moralement désagéable : la dérivée est une notion purement locale, en ce sens que la dérivée de f au point x_i ne dépend que des valeurs prises par f *au voisinage* de x_i, alors que la fonction spline dépend globalement de f par l'intermédiaire d'un système linéaire qui fait intervenir *toutes* les valeurs de f.

On préfère donc le plus souvent utiliser une idée plus simple : on écrit un polynôme d'interpolation au voisinage du point x_i et on dérive celui-ci. Les formules vont varier en fonction du nombre de points qu'on choisit pour écrire le polynôme d'interpolation (en général 2 ou 3, plus rarement 4 ou 5).

3.2.1.1 Formules à deux points

Le polynôme d'interpolation sur les deux points x_i, x_{i+1} s'écrit :

$$P(x) = f(x_i) + f[x_i, x_{i+1}](x - x_i).$$

On a donc $P'(x_i) = f[x_i, x_{i+1}]$, ce qui fournit la

$$\textbf{Formule décentrée à droite} \qquad f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \,. \tag{3.3}$$

On a bien sûr aussi la

$$\textbf{Formule décentrée à gauche} \qquad f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \,. \tag{3.4}$$

3.2.1.2 Formules à trois points

On choisit d'interpoler sur les points x_{i-1}, x_i, x_{i+1} (ce qui est moralement plus satisfaisant). Dans ce cas on a

$$P(x) = f(x_{i-1}) + f[x_{i-1}, x_i](x - x_{i-1}) + f[x_{i-1}, x_i, x_{i+1}](x - x_{i-1})(x - x_i).$$

On a donc

$$P'(x_i) = f[x_{i-1}, x_i] + f[x_{i-1}, x_i, x_{i+1}](x_i - x_{i-1}),$$

ce qui fournit, après simplification la
Formule centrée

$$f'(x_i) \simeq f(x_{i+1}) \frac{h_{i-1}}{h_i(h_{i-1} + h_i)} + f(x_i) \left(\frac{1}{h_{i-1}} - \frac{1}{h_i} \right) - f(x_{i-1}) \frac{h_i}{h_{i-1}(h_{i-1} + h_i)} \,. \tag{3.5}$$

La formule (3.5) se simplifie notablement dans le cas de points équidistants ($h_{i-1} = h_i = h$) pour donner

$$\textbf{Formule centrée - points équidistants} \qquad f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \,. \tag{3.6}$$

Remarquons que la formule ci-dessus n'est autre que la moyenne des deux formules décentrés dans le cas de points équidistants.

3.2.1.3 Erreur

Pour le calcul théorique de l'erreur commise quand on remplace $f'(x_i)$ par l'une des formules approchées ci-dessus, on revient à la démonstration effectuée dans le cadre de l'interpolation. Par exemple, dans le cas de la formule décentrée à gauche, on a, d'après (2.14)

$$f(x) = P(x) + f[x_{i-1}, x_i, x](x - x_{i-1})(x - x_i)$$

d'où, par dérivation et en faisant $x = x_i$

$$|f'(x_i) - P'(x_i)| = f[x_{i-1}, x_i, x_i](x_i - x_{i-1})$$

où $f[x_{i-1}, x_i, x_i]$ est définie comme en (2.36). On obtient finalement, grâce au Lemme 2.1 :

$$\left| f'(x_i) - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \right| \leq \frac{M_2}{2} h_{i-1} \,. \tag{3.7}$$

On démontrerait de même les formules :

$$\left| f'(x_i) - \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right| \leq \frac{M_2}{2} h_i \,. \tag{3.8}$$

et dans le cas centré, avec des points équidistants

$$\left| f'(x_i) - \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \right| \leq \frac{M_3}{6} h^2 \,. \tag{3.9}$$

3.2.2 Dérivées d'ordre supérieur

Le principe est exactement le même, il faut simplement prendre garde que le degré du polynôme d'interpolation soit suffisant pour que sa dérivée n-ième soit non nulle !
Par exemple, pour la dérivée seconde, on choisit en général d'interpoler sur 3 points, ce qui donne (dans le cas de points équidistants)

Dérivée seconde - points équidistants
$$f''(x_i) \simeq \frac{f(x_{i+1}) + f(x_{i-1}) - 2f(x_i)}{h^2} \,. \tag{3.10}$$

avec une erreur majorée par

$$\left| f''(x_i) - \frac{f(x_{i+1}) + f(x_{i-1}) - 2f(x_i)}{h^2} \right| \leq \frac{M_4}{12} h^2 \,. \tag{3.11}$$

3.2.3 Méthode des différences finies

3.2.3.1 Principe de la méthode

La méthode est assez simple à comprendre. Nous allons l'expliquer pour des problèmes en dimension 1 d'espace. Supposons tout d'abord qu'on souhaite résoudre l'équation différentielle ordinaire (ou EDO)

$$\begin{cases} y'' + p(x)\, y' + q(x)\, y = f(x) \quad x \in (0, L) \\ y(0) = a, \quad y(L) = b \end{cases} \tag{3.12}$$

où p, q, f sont des fonctions connues (qu'on supposera continues) et a, b des réels. Le principe de la méthode des différences finies est alors de faire une discrétisation de l'intervalle $(0, L)$ sur lequel est posé l'équation (on choisira en général un pas fixe pour simplifier). Les inconnues deviennent les valeurs de la solution $y(x)$ en chaque point de la grille. On écrit alors, en chaque point intérieur de la grille, les relations liant ces valeurs inconnues en remplaçant dans l'EDO chaque dérivée par des différences à l'aide de l'une des formules (3.6), (3.10)... ci-dessus.

Plus précisément, notons h le pas de discrétisation et $x_0 = 0, x_1 = h, x_2 = 2h, \ldots, x_i = ih, \ldots, x_{N+1} = L = (N+1)h$ les points de la grille. Le but est maintenant de calculer une valeur approchée y_i de la solution $y(x)$ au point x_i. Nous noterons également $p_i = p(x_i), q_i = q(x_i)$ et $f_i = f(x_i)$ les valeurs connues des données aux mêmes points x_i. Maintenant, si on remplace $y''(x_i)$ par la formule (3.10) et $y'(x_i)$ par la formule (3.6) pour tous les points intérieurs correspondant à $i = 1, 2, \ldots N$, on est conduit au système de N équations à N inconnues

$$\frac{y_{i+1} + y_{i-1} - 2y_i}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = f_i \quad i = 1, 2, \ldots N. \tag{3.13}$$

Notons que pour $i = 1$, y_0 peut être remplacé par a, tandis que pour $i = N$, y_{N+1} peut être remplacé par b, par les conditions au bord de l'équation (3.12). Il s'agit d'un système linéaire de N équations à N inconnues (N étant typiquement de l'ordre de plusieurs centaines) qu'on peut écrire matriciellement $AY = F$ où la matrice A et les vecteurs Y et F sont donnés par :

$$A = \begin{pmatrix} q_1 - \frac{2}{h^2} & \frac{1}{h^2} + \frac{p_1}{2h} & 0 & \cdots & 0 \\ \frac{1}{h^2} - \frac{p_2}{2h} & q_2 - \frac{2}{h^2} & \frac{1}{h^2} + \frac{p_2}{2h} & \ddots & \vdots \\ 0 & \frac{1}{h^2} - \frac{p_3}{2h} & q_3 - \frac{2}{h^2} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \frac{1}{h^2} + \frac{p_{N-1}}{2h} \\ 0 & \cdots & 0 & \frac{1}{h^2} - \frac{p_N}{2h} & q_N - \frac{2}{h^2} \end{pmatrix} \tag{3.14}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} \qquad F = \begin{pmatrix} f_1 - a\left(\frac{1}{h^2} - \frac{p_1}{2h}\right) \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \\ f_N - b\left(\frac{1}{h^2} + \frac{p_N}{2h}\right) \end{pmatrix} \tag{3.15}$$

La matrice A est **tridiagonale**, c'est-à-dire qu'elle n'a que trois diagonales d'éléments non nuls et vous verrez au chapitre 6 et 7 des méthodes particulières pour résoudre ce genre de système.

3.2.3.2 Autre exemple : l'équation de la chaleur

Nous considérons maintenant l'équation de la chaleur mono-dimensionnelle :

$$\begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} = f(x,t) & x \in]0, L[, t \in]0, T] \\ u(0,t) = u(L,t) = 0 & \text{conditions au bord} \\ u(x,0) = u_0(x) & \text{condition initiale} \end{cases} \tag{3.16}$$

et on suppose évidemment la donnée initiale u_0 connue. On introduit un pas d'espace h et un pas de temps k, c'est-à-dire qu'on discrétise de manière régulière les deux intervalles de temps et d'espace :

$$0 = x_0 < x_1 < \ldots < x_{i-1} < x_i = ih < x_{i+1} < \ldots < x_p < x_{p+1} = L$$

et

$$0 = t_0 < t_1 < \ldots < t_{n-1} < t_n = nk < t_{n+1} < \ldots < t_q = T\ .$$

Le but est maintenant de calculer une valeur approchée u_i^n de la solution u au point (x_i, t_n) : $u_i^n \simeq u(x_i, t_n)$ (on notera de même $f_i^n = f(x_i, t_n)$). A cet effet, on remplace dans l'équation aux dérivées partielles (3.16) les dérivées par des différences finies du type (3.3), (3.4) pour les dérivées en temps et (3.10) pour la dérivée en espace. Ce qui donne pour la dérivée seconde en espace :

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n) \simeq \frac{u(x_{i+1}, t_n) + u(x_{i-1}, t_n) - 2u(x_i, t_n)}{h^2} \simeq \frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{h^2}\ . \tag{3.17}$$

Pour la dérivée première en temps, on peut utiliser les deux formules (3.3) et (3.4) qui vont conduire à deux schémas complètement différents (notons que la formule (3.6) est non seulement inutilisable ici car on ne peut pas l'initialiser, mais aussi très mauvaise d'un point de vue numérique) :

formule décentrée aval

$$\frac{\partial u}{\partial t}(x_i, t_n) \simeq \frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{k} \simeq \frac{u_i^{n+1} - u_i^n}{k}\,. \tag{3.18}$$

formule décentrée amont

$$\frac{\partial u}{\partial t}(x_i, t_n) \simeq \frac{u(x_i, t_n) - u(x_i, t_{n-1})}{k} \simeq \frac{u_i^n - u_i^{n-1}}{k}\,. \tag{3.19}$$

La formule (3.18) conduit donc au schéma suivant :

$$\textbf{Schéma explicite :} \qquad \frac{u_i^{n+1} - u_i^n}{k} - \frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{h^2} = f_i^n\,. \tag{3.20}$$

On appelle ce schéma **explicite** puisqu'on peut obtenir directement les valeurs u_i^{n+1} à partir des valeurs u_i^n qu'on a calculées à l'étape précédente. Le schéma est évidemment

initialisé à l'aide de la donnée initiale qui fournit les $u_i^0 := u_0(x_i)$. Réécrivons (3.20) sous forme matricielle. Introduisons à cet effet la matrice A et les vecteurs U^n, F^n suivants :

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \qquad U^n = \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ \vdots \\ u_p^n \end{pmatrix} \qquad F^n = \begin{pmatrix} f_1^n \\ f_2^n \\ \vdots \\ \vdots \\ f_p^n \end{pmatrix} \tag{3.21}$$

qui permettent de réécrire (3.20) sous la forme

$$\frac{U^{n+1} - U^n}{k} + \frac{1}{h^2} AU^n = F^n \tag{3.22}$$

ou encore

$$U^{n+1} = \left(I - \frac{k}{h^2} A \right) U^n + kF^n \, . \tag{3.23}$$

(où I est la matrice Identité). C'est évidemment sous cette forme qu'on programmera ce schéma en Matlab. Remarquons que dans le cas $f = 0$, on a l'expression encore plus directe :

$$U^n = \left(I - \frac{k}{h^2} A \right)^n U^0 \, . \tag{3.24}$$

Si on reprend maintenant la formule (3.4) pour discrétiser la dérivée en temps, on obtient le schéma

$$\textbf{Schéma implicite :} \qquad \frac{u_i^n - u_i^{n-1}}{k} - \frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{h^2} = f_i^n \, . \tag{3.25}$$

Soit, en réutilisant la notation matricielle bien plus pratique :

$$\frac{U^n - U^{n-1}}{k} + \frac{1}{h^2} AU^n = F^n \tag{3.26}$$

ou encore

$$\left(I + \frac{k}{h^2} A \right) U^n = U^{n-1} + kF^n \, . \tag{3.27}$$

On appelle ce schéma **implicite** puisque pour obtenir les valeurs u_i^n à partir des valeurs u_i^{n-1} calculées à l'étape précédente, on doit résoudre (à chaque pas de temps) un système linéaire de matrice $I + \frac{k}{h^2} A$.

3.2.3.3 Stabilité

Le choix des pas de temps h et k pour la méthode explicite est absolument crucial. Je représente dans la figure 3.1, les 2 situations où on applique le schéma explicite avec respectivement $k = 0.0050$, $h = 0.1$ et $k = 0.00505$, $h = 0.1$ (la température étant calculée au temps $T = 1$). On aperçoit très nettement de minuscules oscillations qui apparaissent pour la valeur critique $k/h^2 = 0.5$ et ces oscillations ne font que s'amplifier au fur et à mesure

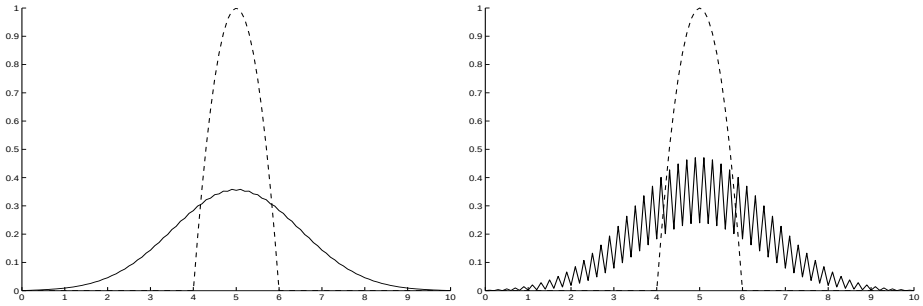

FIGURE 3.1 – À gauche, le schéma explicite avec $k/h^2 = 0.5$, à droite $k/h^2 = 0.505$ (en pointillé la donnée initiale)

que le rapport k/h^2 croît. Par exemple, pour $k/h^2 = 0.55$, on obtient déjà des valeurs de la solution en 10^{13} qui n'ont donc absolument aucun sens !

Ce phénomène d'instabilité est connu depuis longtemps (cf la condition CFL ci-dessous). Donnons-en une explication heuristique. Écrivons le schéma explicite (3.20)

$$\frac{u_i^{n+1} - u_i^n}{k} = \frac{u_{i+1}^n + u_{i-1}^n - 2u_i^n}{h^2}$$

sous la forme

$$u_i^{n+1} = \frac{k}{h^2}\, u_{i-1}^n + (1 - \frac{2k}{h^2})u_i^n + \frac{k}{h^2}\, u_{i+1}^n \,. \tag{3.28}$$

On voit bien que si

$$1 - \frac{2k}{h^2} \geq 0 \;\Leftrightarrow\; k/h^2 \leq 0.5$$

u_i^{n+1} s'écrit comme une **combinaison convexe** des trois valeurs u_{i-1}^n, u_i^n et u_{i+1}^n. En particulier, puisque la donnée initiale est nécessairement bornée : $m \leq u_i^0 \leq M$, on aura par une récurrence immédiate :

$$\forall n,\; m \leq u_i^n \leq M$$

(c'est ce qu'on appelle une version discrète du principe du maximum), ce qui montre que le schéma ne peut exploser et qui est la notion de stabilité qui va être explicitée ci-dessous.

En revanche, supposons la condition $k/h^2 \leq 0.5$ (ou condition CFL) non satisfaite et choisissons pour donnée initiale $u_i^0 = (-1)^i$ (qui est bornée !). On vérifie facilement, par récurrence, que

$$\forall n,\, \forall i\; u_i^n = (-1)^i \left(1 - \frac{4k}{h^2}\right)^n \,.$$

Mais puisque $1 - \frac{4k}{h^2} > 1$, on voit bien que u_i^n explose quand $n \to +\infty$ avec changement de signe comme le laissaient prévoir les expérimentations numériques.

Définition de la stabilité

Les vecteurs U^n que nous manipulons ici sont dans $\mathbb{R}^p$ où la dimension p est manifestement liée au pas de discrétisation spatial (puisque $ph = L$, longueur de l'intervalle). Nous allons

travailler dans la suite avec la norme du max :

$$\|U^n\|_\infty := \max_{1\leq i\leq p} |u_i^n|$$

Définition 3.1 *Un schéma aux différences finies est dit stable pour la norme du max* $\|\ \|_\infty$ *s'il existe une constante* $K > 0$ *indépendante de* k *et* h *(quand ceux-ci tendent vers 0) telle que*

$$\|U^n\|_\infty \leq K\|U^0\| \text{ pour tout } n \tag{3.29}$$

et ce, quelle que soit la donnée initiale U^0.
Si (3.29) n'a lieu que pour des pas k *et* h *astreints à certaines inégalités, on dit que le schéma est conditionnellement stable.*

On montre alors (et on l'admettra ici)

Théorème 3.1 (Condition CFL) *Le schéma explicite (3.20)-(3.24) est stable pour la norme* $\|\ \|_\infty$ *si et seulement si la* **condition CFL** $2k \leq h^2$ *est satisfaite.*
Le schéma implicite (3.25)-(3.27) est stable pour la norme $\|\ \|_\infty$ *quel que soient les pas de temps et d'espace (on dit qu'il est inconditionnellement stable).*

Le nom CFL vient de Courant, Friedrichs et Lewy qui, en 1928 (donc bien avant l'existence des ordinateurs !) ont mis en évidence ce phénomène.

3.3 Intégration numérique : méthodes composites

3.3.1 Principe

On veut calculer $\int_a^b f(x)\,dx$. On décompose l'intervalle $[a,b]$ en $a = x_0 < x_1 < ... < x_n = b$. On a alors

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx.$$

Sur chaque $[x_{i-1}, x_i]$, on applique une méthode d'intégration élémentaire consistant à remplacer $f(x)$ par

$$\begin{aligned} P_i(x) &= f[x_{i,0}] + f[x_{i,0}, x_{i,1}](x - x_{i,0}) + \dots && (3.30)\\ &\quad + f[x_{i,0}, x_{i,1}, ..., x_{i,l}](x - x_{i,0})...(x - x_{i,l}), && (3.31)\end{aligned}$$

soit le polynôme d'interpolation de f en des points $x_{i,0}, ..., x_{i,l}$ de l'intervalle $[a,b]$ (qui peuvent être ou non dans $[x_{i-1}, x_i]$).

3.3.2 Méthode des rectangles

On prend $l = 0$: on approche f par une constante $f(x_{i,0})$ où $x_{i,0} \in [x_{i-1}, x_i]$. D'où la formule de quadrature (i.e. intégration numérique) élémentaire :

$$\int_{x_{i-1}}^{x_i} f(x)\,dx \simeq h_i f(\zeta_i), \quad \zeta_i \in [x_{i-1}, x_i], \quad h_i = x_i - x_{i-1} \tag{3.32}$$

et la formule de quadrature composée :

$$\int_a^b f(x)\,dx \simeq \sum_{i=1}^{n} h_i f(\zeta_i)\,. \tag{3.33}$$

On reconnait là une somme de Riemann dont on sait qu'elle converge vers $\int_a^b f(x)\,dx$ quand $\max_{1\leq i\leq n} h_i \longrightarrow 0$ si f est Riemann-intégrable. Les choix courants pour ζ_i sont :

- rectangles à gauche : $\zeta_i = x_{i-1} \;\rightarrow\; \int_a^b f(x)\,dx \simeq \sum_{i=1}^{n} h_i f(x_{i-1})$
- rectangles à droite : $\zeta_i = x_i \;\rightarrow\; \int_a^b f(x)\,dx \simeq \sum_{i=1}^{n} h_i f(x_i)$
- formule du point milieu : $\zeta_i = \dfrac{x_{i-1}+x_i}{2} (:= x_{i-\frac{1}{2}}) \;\rightarrow\; \displaystyle\int_a^b f(x)\,dx \simeq \sum_{i=1}^{n} h_i f\left(x_{i-\frac{1}{2}}\right)$

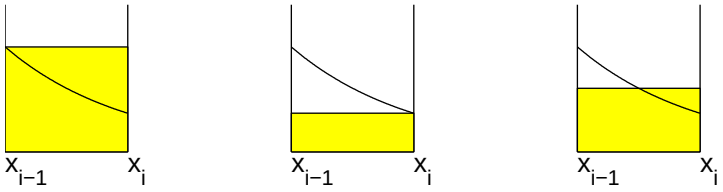

FIGURE 3.2 – Formule des rectangles à gauche, des rectangles à droite, du point milieu

La figure 3.2 suggère que la formule du point milieu doit fournir la meilleure approximation des trois en général (voir le paragraphe sur le calcul d'erreur).

3.3.3 Méthode des trapèzes

On prend $l = 1$ et on remplace f par son interpolé linéaire aux points $[x_{i-1}, x_i]$.

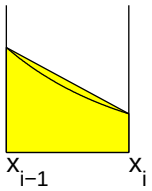

FIGURE 3.3 – Méthode des trapèzes

$$\int_{x_{i-1}}^{x_i} f(x)\,dx \simeq \frac{1}{2}(x_i - x_{i-1})(f(x_i) + f(x_{i-1})).$$

$$\int_a^b f(x)\,dx \simeq \sum_{i=1}^{n-1} \frac{h_i + h_{i+1}}{2} f(x_i) + \frac{1}{2}(h_1 f(x_0) + h_n f(x_n))$$

et dans le cas d'une subdivision régulière ($h_i = h,\ \forall i = 1, ..., n$) :

$$\int_a^b f(x)\,dx \simeq h\left[\sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}(f(x_0) + f(x_n))\right].$$

3.3.4 Méthode de Simpson

Cette fois on interpole f aux points x_{i-1}, x_i et $x_{i-\frac{1}{2}} = \frac{x_{i-1}+x_i}{2}$ soit, en utilisant la formule d'interpolation de Newton (2.5) :

$$\begin{aligned}\int_{x_{i-1}}^{x_i} f(x)\,dx &\simeq \int_{x_{i-1}}^{x_i} \left[f(x_{i-1}) + f[x_{i-1}, x_i](x - x_{i-1}) + f\left[x_{i-1}, x_i, x_{i-\frac{1}{2}}\right](x - x_{i-1})(x - x_i)\right] dx \\ &= h_i f(x_{i-1}) + \frac{1}{2}h_i^2 f[x_{i-1}, x_i] + f\left[x_{i-1}, x_i, x_{i-\frac{1}{2}}\right] \int_{x_{i-1}}^{x_i} (x - x_{i-1})(x - x_i)\,dx\end{aligned}$$

or

$$\int_{x_{i-1}}^{x_i} (x - x_{i-1})(x - x_i)\,dx = \frac{1}{3}h_i^3 - \frac{1}{2}h_i^3 = -\frac{h_i^3}{6} \quad \text{(en écrivant } x - x_i = x - x_{i-1} + x_{i-1} - x_i)$$

et

$$h_i^2 f[x_{i-1}, x_i] = h_i(f(x_i) - f(x_{i-1}))$$

$$\begin{aligned}h_i^3 f\left[x_{i-1}, x_{i-\frac{1}{2}}, x_i\right] &= h_i^2 \left(\frac{f(x_i) - f\left(x_{i-\frac{1}{2}}\right)}{\frac{h_i}{2}} - \frac{f\left(x_{i-\frac{1}{2}}\right) - f(x_{i-1})}{\frac{h_i}{2}}\right) \\ &= 2h_i\left(f(x_i) - 2f\left(x_{i-\frac{1}{2}}\right) + f(x_{i-1})\right).\end{aligned}$$

On obtient donc

$$\int_{x_{i-1}}^{x_i} f(x)\,dx = \frac{h_i}{6}\left(f(x_{i-1}) + 4f\left(x_{i-\frac{1}{2}}\right) + f(x_i)\right).$$

La formule composée dans le cas de pas constants devient

$$\int_a^b f(x)\,dx = \frac{h}{6}\left[f_0 + f_n + 2\sum_{i=1}^{n-1} f_i + 4\sum_{i=1}^{n} f_{i-\frac{1}{2}}\right]$$

où on note $f_i = f(x_i)$, $f_{i-\frac{1}{2}} := f\left(x_{i-\frac{1}{2}}\right) = f\left(\frac{x_{i-1}+x_i}{2}\right)$.

Remarque 3.1 Pour cette méthode, il est nécessaire de connaitre f aux points x_i et en leurs milieux, soit en un nombre impair de points. De manière générale, on appelle Formules de Newton-Cotes (fermées) les formules composées lorsque, sur chaque intervalle $[x_{i-1}, x_i]$, f est interpolée en l points équidistants. Les cas $l = 1$ et $l = 2$ correspondent donc respectivement aux formules des trapèzes et de Simpson, le cas $l = 4$ est connu dans la littérature sous le nom de Boole-Villarceau.

3.4 Analyse de l'erreur dans les méthodes d'intégration

3.4.1 Théorie

Nous évaluons ici l'erreur faite lorsqu'on remplace $\int_a^b f(x)dx$ par $\int_a^b P(x)dx$ où P est le polynôme d'interpolation de f aux points $x_0, ..., x_N$.
On a vu au chapitre précédent que si $P_N(x)$ est le polynôme d'interpolation de f en $x_0, x_1, ..., x_N$ alors

$$\begin{aligned} f(x) - P_N(x) &= f[x_0, x_1, ..., x_N, x]\,\psi_N(x) && (3.34)\\ \text{avec } \psi_N(x) &= (x-x_0)(x-x_1)...(x-x_N). && (3.35)\end{aligned}$$

Ainsi

$$E(f) = \int_a^b f(x)\,dx - \int_a^b P_N(x)\,dx = \int_a^b f[x_0, x_1, ..., x_N, x]\,\psi_N(x)\,. \tag{3.36}$$

L'expression de $E(f)$ se simplifie dans deux cas particuliers.

A $\psi_N(x)$ est de signe constant sur $[a,b]$: on peut alors utiliser le théorème de la moyenne pour les intégrales qui affirme que, si g et h sont continues sur $[a,b]$ et si $h(x) \geq 0$ sur $[a,b]$, il existe $\zeta \in [a,b]$ tel que

$$\int_a^b g(x)\,h(x)\,dx = g(\zeta)\int_a^b h(x)\,dx.$$

Appliqué à (3.36), ceci montre l'existence de $\zeta \in [a,b]$ tel que

$$E(f) = f[x_0, x_1, ..., x_N, \zeta]\int_a^b \psi_N(x)\,dx.$$

D'après le lemme 2.1 du chapitre précédent, on en déduit encore qu'il existe $\eta \in [c,d]$ ($[c,d]$ est un intervalle contenant les x_i, a et b) tel que

$$E(f) = \frac{1}{(N+1)!} f^{(N+1)}(\eta)\int_a^b \psi_N(x)\,dx. \tag{3.37}$$

(On supposera ici et dans toute la suite que f est assez régulière).

B $\int_a^b \psi_N(x)\,dx = 0$; dans ce cas, on utilise

$$f[x_0, x_1, ..., x_N, x] = f[x_0, x_1, ..., x_N, x_{N+1}] + (x_{N+1} - x)\,f[x_0, x_1, ..., x_N, x_{N+1}, x] \tag{3.38}$$

valable pour x_{N+1} arbitraire. Alors, (3.36) devient

$$E(f) = \int_a^b f[x_0, x_1, ..., x_N, x_{N+1}]\,\psi_N(x)\,dx + \int_a^b f[x_0, x_1, ..., x_N, x_{N+1}, x]\,\psi_{N+1}(x)\,dx$$

$$E(f) = \int_a^b f[x_0, x_1, ..., x_N, x_{N+1}, x]\,\psi_{N+1}(x)\,dx. \tag{3.39}$$

Si on peut choisir x_{N+1} de telle façon que $\psi_{N+1}(x)$ reste de signe constant sur $[a,b]$, on aura d'après (3.37)

$$E(f) = \frac{1}{(N+2)!} f^{(N+2)}(\eta) \int_a^b \psi_{N+1}(x)\,dx \tag{3.40}$$

$$\text{pour un } \eta \in [c,d] \text{ (intervalle contenant les } x_i, a \text{ et } b) \tag{3.41}$$

Définition 3.2 *On dit qu'une méthode d'intégration numérique est d'ordre k si elle est exacte pour tous les polynômes de degré inférieur ou égal à k.*

Remarque 3.2 Lorsque $\psi_N(x)$ est de signe constant sur $[a,b]$, d'après (3.37), on a une méthode d'ordre N puisque $P^{(N+1)}(\eta) = 0$ pour tout polynôme de degré inférieur ou égal à N. Dans la situation B, si (3.37) est vraie, on aura même une méthode d'ordre $N+1$ (avec un même nombre de points).

3.4.2 Application aux méthodes usuelles

3.4.2.1 Méthode des rectangles (à gauche ou à droite)

$N = 0$, $x_0 = a$, $\psi_0(x) = x - a$. On peut appliquer (3.37) qui donne

$$E(f) = f'(\eta) \int_a^b (x-a)\,dx = \frac{(b-a)^2}{2} f'(\eta).$$

La méthode est d'ordre 0 et exacte seulement pour les constantes.

$$\left| \int_{x_{i-1}}^{x_i} f(x)\,dx - h_i f(x_{i-1}) \right| \leq \frac{h_i^2}{2} f'(\eta_i) \leq \frac{h_i^2}{2} M_1 \tag{3.42}$$

$$\text{où } M_1 := \max_{a \leq x \leq b} \left| f'(x) \right| \tag{3.43}$$

Pour la formule composée, il faut ajouter les erreurs provenant de chaque intégration élémentaire. Dans le cas d'un pas constant, on obtient :

$$\left| \int_a^b f(x)\,dx - h \sum_{i=1}^{N} f(x_{i-1}) \right| \leq \frac{M_1}{2} (b-a)\, h.$$

3.4.2.2 Formule du point milieu

$$N = 0, \quad x_0 = \frac{a+b}{2} \quad \psi_0(x) = x - \frac{a+b}{2}.$$

Ici, $\int_a^b \psi_0(x)\,dx = 0$. On va donc obtenir une méthode d'ordre 1 (exacte pour toute fonction linéaire) bien qu'on interpole par une constante. En effet, d'après (3.40) appliqué avec $x_{N+1} = x_0 = x_1 = \frac{a+b}{2}$

$$\left| \int_a^b f(x)\,dx - \frac{b-a}{2} (f(a) + f(b)) \right| \leq \frac{M_2}{2} \int_a^b \left(x - \frac{b+a}{2} \right)^2 dx = \frac{M_2}{24} (b-a)^3$$

avec $M_2 := \max_{a\leq x\leq b} |f''(x)|$. Pour la formule composée associée, on obtient

$$\left| \int_a^b f(x)\,dx - h\sum_{i=1}^{N} f\left(x_{i-1/2}\right) \right| \leq \frac{M_2}{24} h^2 (b-a).$$

3.4.2.3 Méthode des trapèzes

$$N=1, \quad x_0 = a,\ x_1 = b \quad \psi_1(x) = (x-a)(x-b).$$

Comme ψ_1 est de signe constant sur $[a,b]$, on applique (3.37) pour trouver

$$\left| \int_a^b f(x)\,dx - \frac{(b-a)}{2}(f(a)+f(b)) \right| \leq \frac{|f''(\eta)|}{2} \int_a^b (x-a)(x-b)\,dx$$

soit

$$\left| \int_a^b f(x)\,dx - \frac{(b-a)}{2}(f(a)+f(b)) \right| \leq \frac{M_2}{12}(b-a)^3.$$

Pour la formule composée de la méthode des trapèzes, on en déduit que l'erreur est majorée par

$$E(f) \leq \frac{M_2}{12} h^2 (b-a). \tag{3.44}$$

3.4.2.4 Méthode de Simpson

$$N=2, \quad x_0 = a,\ x_1 = \frac{a+b}{2},\ x_2 = b \quad \psi_2(x) = (x-a)\left(x - \frac{a+b}{2}\right)(x-b).$$

Par raison de symétrie, on vérifie que :

$$\int_a^b \psi_2(x)\,dx = 0.$$

Comme $\psi_3(x) = (x-a)\left(x-\frac{a+b}{2}\right)^2 (x-b)$ est de signe constant sur $[a,b]$, on peut appliquer (3.40) pour obtenir

$$\left| \int_a^b f(x)\,dx - \frac{b-a}{6}\left(f(a) + 4f\left(\frac{b-a}{2}\right) + f(b)\right) \right| \leq \frac{M_4}{24} \int_a^b \psi_3(x)\,dx$$

soit tous calculs faits

$$|E(f)| \leq \frac{M_4}{90}\left(\frac{b-a}{2}\right)^5 \tag{3.45}$$

$$\left(M_4 = \max_{a\leq x\leq b} \left|f^{(4)}(x)\right|\right). \tag{3.46}$$

La formule composée correspondante avec pas constant h donnera une erreur majorée par

$$|E(f)| \leq \frac{M_4}{2880} h^4 (b-a).$$

Par ailleurs, la méthode de Simpson (bien que reposant sur une interpolation à trois points) est exacte pour tout polynôme de degré inférieur ou égal à 3.

3.4.3 Exemple d'application

Calcul de $\int_0^1 e^{-x^2}dx$.
Utilisons les méthodes élémentaires précédentes à l'aide des valeurs de $f(x) = e^{-x^2}$ aux points 0, $\frac{1}{2}$, 1 soit

$$f(0) = 1,\ f\left(\frac{1}{2}\right) = 0,77880,\ f(1) = e^{-1} = 0,36788.$$

D'où les résultats

	rectangle	point milieu	trapèzes	Simpson
$\int_0^1 e^{-x^2}dx$	1	0,77880	0,68394	0,74718
erreur	0,25318	0,03198	−0,06288	0,00036

La valeur exacte est 0,74682. Ces résultats sont tout à fait en accord avec les considérations théoriques précédentes. Noter comment la méthode de Simpson donne une approximation à 4.10^{-4} avec seulement 3 valeurs de f ! Ceci est bien sûr dû au fait que les dérivées d'ordre supérieur ne varient pas trop sur l'intervalle $[0,1]$.

3.5 Méthodes d'intégration numérique de Gauss

3.5.1 Introduction

Les méthodes composées décrites précédemment consistent à remplacer $\int_a^b f(x)\,dx$ par une expression de la forme

$$A_0 f(x_0) + A_1 f(x_1) + ... + A_N f(x_N)$$

où les coefficients A_i sont indépendants de f et où les x_i sont N points de $[a,b]$. Au moins dans les exemples considérés (et c'est un fait général), les méthodes de type Newton-Cotes sont exactes pour les polynômes de degré inférieur ou égal à N (parfois même de degré $N+1$). Il est en fait possible avec le même nombre de points d'obtenir des formules exactes pour des polynômes de degré inférieur ou égal à $2N+1$: c'est le cas des méthodes de Gauss. Outre cet avantage, elles permettent également de calculer des intégrales singulières où la fonction f devient infinie en certains points par exemple

$$\int_0^1 \frac{e^{-x^2}}{\sqrt{x}}dx.$$

Pour recouvrir ce type de situation, on va considérer plus généralement des intégrales de la forme

$$\int_a^b g(x)\,w(x)\,dx$$

où w est une fonction poids qui, dans la pratique, contiendra la singularité (ainsi $w(x) = \frac{1}{\sqrt{x}}$ dans l'exemple ci-dessus).

3.5.2 Idée de base des méthodes de Gauss

Soit $x_0, ..., x_N \in [a,b]$ et $P_N(x)$ le polynôme d'interpolation de g aux points x_i. On a

$$g(x) = P_N(x) + g[x_0, x_1, ..., x_N, x]\,\psi_N(x) \tag{3.47}$$
$$\psi_N(x) = (x-x_0)(x-x_1)...(x-x_N). \tag{3.48}$$

Ceci donne en posant $I(g) = \int_a^b g(x)\,w(x)\,dx$,

$$I(g) - I(P_N) = \int_a^b g[x_0, x_1, ..., x_N, x]\,\psi_N(x)\,w(x)\,dx.$$

Supposons que

$$\int_a^b \psi_N(x)\,w(x)\,dx = 0.$$

Alors, raisonnant comme en (3.38), (3.39) et utilisant

$$g[x_0, x_1, ..., x_N, x] = g[x_0, x_1, ..., x_N, x_{N+1}] + (x - x_{N+1})\,g[x_0, x_1, ..., x_N, x_{N+1}, x]$$

pour x_{N+1} quelconque, on obtient avec $\psi_{N+1}(x) = \psi_N(x)(x - x_{N+1})$.

$$I(g) - I(P_N) = \int_a^b g[x_0, x_1, ..., x_N, x_{N+1}, x]\,\psi_{N+1}(x)\,w(x)\,dx.$$

Si de même

$$\int_a^b \psi_{N+1}(x)\,w(x)\,dx = 0$$

on peut répéter le procédé avec un nouveau point x_{N+2}. De façon générale si

$$\int_a^b \psi_N(x)(x - x_{N+1})...(x - x_{N+m})\,w(x)\,dx = 0,\ m = 0, 1, \ldots \tag{3.49}$$

on obtient à l'aide d'un nouveau point x_{N+m+1}

$$I(g) - I(P_N) = \int_a^b g[x_0, x_1, ..., x_N, x_{N+1}, ..., x_{N+m+1}, x]\,\psi_{N+m+1}(x)\,w(x)\,dx. \tag{3.50}$$

L'idée est de choisir les points $x_0, ...x_N$ initiaux de telle façon que le polynôme de degré $N+1$, $\psi_N(x)$ satisfasse (3.49), c'est-à-dire soit orthogonal au polynôme $(x - x_{N+1})...(x - x_{N+m})$ pour le produit scalaire associé au poids w.
Ainsi, si $P_0, P_1, ..., P_N, P_{N+1}$ est la suite de polynômes orthogonaux (normalisés) associés à ce produit scalaire, on sait déjà d'après la proposition 2.1 du chapitre précédent que :

$$(i)\ P_{N+1} = (x - \zeta_0)(x - \zeta_1)...(x - \zeta_N)$$

où les ζ_i sont réels et distincts :

$$(ii) \int_a^b P_{N+1}(x)\,q(x)\,w(x)\,dx = 0$$

pour tout polynôme q avec $\deg q \leq N$. D'où le choix :

$$\begin{array}{ll} \forall i = 0, 1, ..., N & x_i := \zeta_i \\ \forall i = N\ +1, ..., 2N+1 & x_i := \zeta_{2N+1-i}. \\ & m = N \end{array}$$

On a alors :

$$\psi_N(x) = P_{N+1}(x)$$

et d'après (3.49), (3.50)

$$I(g) - I(P_N) = \int_a^b g[x_0, ..., x_N, x_{N+1}, ..., x_{2N+1}] (P_{N+1}(x))^2 w(x)\, dx. \tag{3.51}$$

Comme $(P_{N+1}(x))^2$ est de signe constant, on peut appliquer le théorème de la moyenne à cette intégrale, puis appliquant le lemme 2.1 du chapitre précédent (comme dans (3.37)), on obtient l'existence de $\zeta \in [a, b]$ tel que :

$$I(g) - I(P_N) = \frac{g^{(2N+2)}(\zeta)}{(2N+2)!} \int_a^b (P_{N+1}(x))^2 w(x)\, dx. \tag{3.52}$$

On obtient donc une méthode d'intégration numérique d'ordre $2N+1$ avec seulement $N+1$ points !...

Calcul de $I(P_N)$:
Si on écrit le polynôme d'interpolation P_N sous sa forme de Lagrange, on obtient :

$$I(P_N) = \sum_{i=0}^{N} g(x_i) A_i, \text{ avec}$$

$$A_i = \int_a^b l_i(x) w(x)\, dx,\ m = N \quad l_i(x) = \prod_{j=0, j\neq i}^{N} \frac{x - x_j}{x_i - x_j}. \tag{3.53}$$

3.5.3 Mise en œuvre de la méthode

Comme application de ce qui précède, examinons tout d'abord le cas du calcul d'une intégrale du type $\int_a^b f(x)w(x)\, dx$ où a et b sont deux réels (finis). Comme les polynômes orthogonaux sont mieux connus sur $[-1, 1]$, on s'y ramène toujours (quand on travaille sur un intervalle borné) à l'aide du changement de variable :

$$x = \frac{b+a}{2} + \frac{b-a}{2} t$$

Ainsi :

$$\int_a^b f(x) w(x)\, dx = \int_{-1}^{1} f(x(t)) w(x(t)) x'(t)\, dt = \frac{b-a}{2} \int_{-1}^{1} f(x(t)) w(x(t)) dt.$$

Ceci étant acquis, il suffit de se limiter aux intégrales du type :

$$I(g) = \int_{-1}^{1} g(x) w(x)\, dx.$$

3.5.3.1 Méthode de Legendre-Gauss

C'est le cas $w(x) \equiv 1$: Les polynômes orthogonaux appropriés sont ceux de Legendre, soit :

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= x^2 - \frac{1}{3},\ \zeta_0 = -\frac{1}{\sqrt{3}},\ \zeta_1 = \frac{1}{\sqrt{3}}. \end{aligned}$$

On a ainsi la formule à deux points ($N = 1$)

$$\begin{aligned} \int_{-1}^{1} g(x)\,dx &\approx A_0 g\left(-\frac{1}{\sqrt{3}}\right) + A_1 g\left(\frac{1}{\sqrt{3}}\right) \\ \text{avec } A_0 &= \int_{-1}^{1} \frac{x - \frac{1}{\sqrt{3}}}{-\frac{2}{\sqrt{3}}} dx = 1 \text{ voir (3.53)} \\ A_1 &= \int_{-1}^{1} \frac{x + \frac{1}{\sqrt{3}}}{\frac{2}{\sqrt{3}}} dx = 1. \end{aligned}$$

Plus généralement, si on utilise N points, la formule de Legendre-Gauss va s'écrire

$$\int_{-1}^{1} g(x)\,dx \approx \sum_{i=0}^{N} w_i g(z_i) \tag{3.54}$$

où les poids w_i et les racines z_i sont donnés par le tableau suivant. La quatrième colonne du tableau fournit **un majorant** de l'erreur (comme d'habitude M_n désigne la norme infinie de la dérivée n-ième de la fonction sur l'intervalle $[-1, 1]$. Voici comment sont établies ces formules d'erreur : par exemple pour deux points. D'après (3.52), l'erreur dans la formule à deux points est donnée par $E = \frac{g^{(4)}(\zeta)}{24} \int_{-1}^{1} \left(x^2 - \frac{1}{3}\right)^2 dx$. On vérifie $\int_{-1}^{1} \left(x^2 - \frac{1}{3}\right)^2 dx = \frac{8}{45}$ d'où $E = \frac{g^{(4)}(\zeta)}{135}$.

N	Racines z_i	Poids w_i	Formule d'erreur
1	± 0.5773502691	1	$\frac{M_4}{135}$
2	0	0.8888888888	$\frac{M_6}{15750}$
	± 0.7745966692	0.5555555555	
3	± 0.3399810435	0.6521451548	$\frac{M_8}{3472875}$
	± 0.8611363115	0.3478548451	
4	0	0.5688888888	$\frac{M_{10}}{1237732650}$
	± 0.5384693101	0.4786286704	
	± 0.9061798459	0.2369268850	
5	± 0.2386191860	0.4679139345	$\frac{M_{12}}{648984486150}$
	± 0.6612093864	0.3607615730	
	± 0.9324695142	0.1713244923	

(3.55)

Appliquons ce calcul à l'exemple déjà traité par les méthodes composées soit $\int_0^1 e^{-x^2}dx$. On pose $x = \frac{1}{2}(1+t)$ et donc

$$\int_0^1 e^{-x^2}dx = \frac{1}{2}\int_{-1}^1 \exp\left(-\frac{(t+1)^2}{4}\right)dt \simeq \frac{1}{2}\left(\exp\left(\frac{-\left(1-\frac{1}{\sqrt{3}}\right)^2}{4}\right)+\exp\left(\frac{-\left(1+\frac{1}{\sqrt{3}}\right)^2}{4}\right)\right).$$

On obtient, pour la formule avec deux points seulement

$$\begin{aligned}\int_0^1 e^{-x^2}dx &\simeq \frac{1}{2}\left(\exp\left(-\frac{\left(\sqrt{3}-1\right)^2}{12}\right)+\exp\left(-\frac{\left(\sqrt{3}+1\right)^2}{12}\right)\right)\\ &\simeq \frac{1}{2}(0,95632+0,53687)\simeq 0,74659.\end{aligned}$$

Le résultat exact est $0,74682$ soit une erreur de $0,00033$ donc bien meilleure que celle des trapèzes ($-0,06288$ avec deux points) et du même ordre que celle de Simpson ($0,00036$ mais avec trois points).

Remarque 3.3 Comme je l'ai déjà signalé plus haut, il est parfaitement possible (en particulier quand l'intervalle d'intégration est grand) de reprendre l'idée des méthodes composites avec des formules de Gauss. Sur chaque sous-intervalle, on fera alors le changement de variable indiqué au début du paragraphe.

3.5.3.2 Méthode de Gauss-Chebyshev

C'est le cas $w(x) = \frac{1}{\sqrt{1-x^2}}$: Les polynômes appropriés sont les polynômes de Chebyshev dont les zéros sont (voir chapitre précédent)

$$\zeta_i = \cos\frac{(2i+1)\pi}{2(N+1)} \quad i = 0, ..., N.$$

Il se trouve que dans ce cas, tous les coefficients A_i sont égaux et on a la formule de quadrature très attrayante :

$$\int_{-1}^1 \frac{g(x)}{\sqrt{x^2-1}} \approx \frac{\pi}{N+1}\sum_{i=0}^N g\left[\cos\frac{(2i+1)\pi}{2(N+1)}\right]. \tag{3.56}$$

3.5.3.3 Méthodes de Laguerre et Hermite

La méthode de Laguerre correspond au cas $w(x) = e^{-x}$ sur l'intervalle $[a,b[= [0,+\infty[$, tandis que la méthode de Hermitee correspond au cas $w(x) = e^{-x^2}$ sur l'intervalle $]a,b[= [-\infty,+\infty[$. Les formules de récurrence permettant de calculer les polynômes orthogonaux correspondants sont donnés à la section 2.2.4. Nous renvoyons à la littérature pour les formules explicites et plus de détails.

3.6 Exercices du chapitre 3

Exercice 3.1 Calculer $\int_1^3 \frac{(\sin x)^2}{x} dx$ en utilisant successivement

a) la méthode des rectangles sur 5 sous-intervalles,

b) la méthode des trapèzes sur 5 sous-intervalles,

c) la méthode de Simpson sur 5 sous-intervalles,

d) la méthode de Gauss à 5 points

Comparer les résultats et le nombre d'évaluations de fonctions nécessaires par chaque méthode.

Exercice 3.2 Déterminer le nombre de points nécessaires pour évaluer l'intégrale $\int_0^1 \cos(x^2)\, dx$ avec une précision de 10^{-8} par la méthode des rectangles à gauche, la méthode des trapèzes, la méthode de Simpson et la méthode de Gauss composite à 3 points.

Exercice 3.3 Imaginer une méthode de Gauss pour calculer les intégrales

$$\int_0^1 (\cos x)^\alpha \log x \, dx \text{ pour } \alpha = \frac{1}{2}, 1, \frac{3}{2} 2, \frac{5}{2}.$$

Déterminer les premiers polynômes orthogonaux et leurs racines et mettre en oeuvre la méthode de Gauss.

Exercice 3.4 Soit K le carré unité de $\mathbb{R}^2$. Imaginer une formule d'intégration numérique sur K qui soit exacte pour les polynômes de degré inférieur ou égal à 3 en x et en y.

Exercice 3.5 Déterminer les 4 premiers polynômes orthogonaux pour le produit scalaire

$$\int_1^{+\infty} f(t)g(t)e^{-t}\, dt.$$

En déduire une formule d'intégration numérique pour calculer des intégrales du type $\int_1^{+\infty} f(t)e^{-t}\, dt$ qui soit exacte pour les polynômes de degré inférieur ou égal à 7. Tester cette formule avec $f(t) = t^5$, $f(t) = t^6$ et $f(t) = e^{t/2}$.

Exercice 3.6 Calculer $\int_{0.5}^{+\infty} e^{-t^3}\, dt$ avec 4 décimales exactes.

3.7 TD3 : Intégration numérique et méthode des différences finies

Exercice 1
On veut calculer l'intégrale $\int_0^1 \cos(x^2)\,dx$ avec une précision de 10^{-8}. Déterminer le nombre de points nécessaires pour l'évaluer avec cette précision par la méthode des trapèzes et par la méthode de Simpson. Écrire une fonction Matlab pour faire ce calcul avec n points, puis la tester avec les valeurs de n trouvées précedemment. Vérifier avec la fonction *quad* de Matlab.

Exercice 2
Soit l'équation différentielle, avec conditions aux bords de Dirichlet homogènes, suivante :

$$\begin{cases} -y''(x) + p(x)y'(x) + q(x)y(x) = f(x) \\ y(0) = y(1) = 0 \end{cases}$$

On discrétise l'intervalle $[0, 1]$ en $n + 2$ points équidistants :

$$x_k = \frac{k}{n+1} = kh \text{ pour } 0 \leq k \leq n+1$$

On note y_k, p_k, q_k et f_k la valeur des fonctions y, p, q et f au point x_k, $0 \leq k \leq n+1$. On approche alors les dérivées premières et secondes de y par les formules centrées :

$$y'(x_k) = \frac{y_{k+1} - y_{k-1}}{2h} \quad \text{et} \quad y''(x_k) = \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} \quad \text{pour} \quad 1 \leq k \leq n$$

Ce qui permet d'obtenir un système de n équations à n inconnues $y_1, y_2, \cdots, y_n$.

Ecrire la matrice de ce système. Ecrire ensuite une fonction Matlab qui permette de construire cette matrice (on suppose que les fonctions p, q, f sont définies dans des fichiers Matlab à part), puis de résoudre le système associé à l'équation différentielle. Visualiser la solution approchée.
Fonction Matlab utile : diag

Choisir des fonctions p, q et f permettant de connaitre théoriquement la solution de l'équation différentielle. On mesure l'erreur entre la solution théorique $\hat{y}$ et la solution approchée y par :

$$e = \max_{0 \leq k \leq n+1} |y_k - \hat{y}(x_k)|$$

Tracer la variation de e en fonction du pas h.

3.8 Aide-mémoire

Redonnons ici les méthodes usuelles pour calculer une intégrale $\int_a^b f(x)\,dx$ de façon exacte.

3.8.1 Utilisation d'une primitive

Si F est une primitive de f, c'est-à-dire que $F'(x) = f(x)$ alors

$$\int_a^b f(x)\,dx = F(b) - F(a)\,.$$

3.8.2 Intégration par parties

voir Section 1.7.2

3.8.3 Changement de variables

On considère φ une fonction de classe C^1 et soit a_1, b_1 tels que $\varphi(a_1) = a$, $\varphi(b_1) = b$, alors en posant $x = \varphi(t)$, on obtient :

$$\int_a^b f(x)\,dx = \int_{a_1}^{b_1} f(\varphi(t))\,\varphi'(t)\,dt\,.$$

Notez qu'il n'est pas nécessaire que φ soit une bijection. Bien sûr, si elle est bijective, il sera plus facile de déterminer a_1 et b_1 puisqu'il suffira de prendre $a_1 = \varphi^{-1}(a)$ et $a_b = \varphi^{-1}(b)$.

Dans le cours de probabilités du second semestre, vous utiliserez aussi l'analogue en plusieurs dimensions de la formule précédente. Donnons cette formule dès maintenant. Soit U un ouvert de $\mathbb{R}^N$ et $G : U \to \mathbb{R}^N$ une application injective, de classe C^1. Notez bien que G est à valeurs vectorielles, c'est-à-dire que l'image d'un $X = (x_1, x_2, \dots x_N)$ est un vecteur $(g_1(X), g_2(X), \dots, g_N(X))$. On appelle **matrice jacobienne de** G, et on note J_G la matrice de toutes les dérivées partielles premières :

$$J_G(X) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \dots & \frac{\partial g_1}{\partial x_N} \\ \vdots & \vdots & \vdots \\ \frac{\partial g_i}{\partial x_1} & \dots & \frac{\partial g_i}{\partial x_N} \\ \vdots & \vdots & \vdots \\ \frac{\partial g_N}{\partial x_1} & \dots & \frac{\partial g_N}{\partial x_N} \end{pmatrix}$$

(autrement dit, c'est la matrice de terme général $\frac{\partial g_i}{\partial x_j}$). On appelle **Jacobien de** G le déterminant de la matrice jacobienne, on le notera $det(J_G)$. On suppose enfin que le jacobien de G est non nul pour tout x dans U. Posons $V = G(U)$, alors, on a la formule de changement de variable multi-dimensionnel :

$$\int_V f(X)\,dX = \int_U f(G(Y))\,|det(J_G(Y))|\,dY\,.$$

Chapitre 4

Équations différentielles - Théorie et méthodes d'Euler

Il s'agit d'étudier et de résoudre numériquement les systèmes d'équations différentielles du type

$$\begin{cases} y'(t) = f(t, y(t)), \ t > 0 \\ y(0) = y_0 \qquad \text{donné} \end{cases} \tag{4.1}$$

où $y : [0, +\infty[\to \mathbb{R}^N$ est une fonction vectorielle inconnue de la variable t (représentant le plus souvent le temps), y_0 un vecteur donné de $\mathbb{R}^N$ décrivant l'état initial et f une application de $[0, +\infty[\times \mathbb{R}^N$ dans $\mathbb{R}^N$.

4.1 Quelques résultats théoriques sur l'équation différentielle (4.1)

Si $y(t) = (y_1(t), y_2(t), ..., y_N(t))$, le système (4.1) s'écrit

$$\begin{cases} y_1'(t) = f(t, y_1(t), ..., y_N(t)) \\ y_2'(t) = f(t, y_1(t), ..., y_N(t)) \\ \vdots \\ y_N'(t) = f(t, y_1(t), ..., y_N(t)) \end{cases}, \quad y_i(0) = y_0^i \qquad i = 1, ..., N. \tag{4.2}$$

La discrétisation spatiale d'équations aux dérivées partielles modélisant les phénomènes d'évolution est une source importante de tels problèmes. Ils apparaissent aussi souvent directement sous la forme (4.2) dans les applications. Notons qu'une équation d'ordre strictement supérieur à 1 peut-être aussi écrite sous la forme (4.2). Considérons par exemple le problème d'ordre 2 :

$$\begin{cases} y''(t) = g(t, y(t), y'(t)) \\ y(0) = y_0, \ y'(0) = y_1 \end{cases} \tag{4.3}$$

où $y : [0, +\infty[\to \mathbb{R}$ est la fonction inconnue, $g : [0, +\infty[\times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, y_0 et y_1 sont des réels donnés (position et vitesse initiales). Posant alors

$$u_1(t) = y(t), \ u_2(t) = y'(t),$$

on remplace (4.3) par un système équivalent où l'inconnue est le vecteur $U(t) = \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$ soit

$$\frac{dU}{dt}(t) = \begin{pmatrix} u_2(t) \\ g(t, u_1(t), u_2(t)) \end{pmatrix}, \; U(0) = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$$

ce qui est un système de la forme (4.2) avec $N = 2$ et $f(t, u_1, u_2) = \begin{pmatrix} u_2 \\ g(t, u_1, u_2) \end{pmatrix}$.

La même remarque vaut pour les équations d'ordre p

$$\begin{cases} y^{(p)}(t) = g\left(t, y(t), ..., y^{p-1}(t)\right) \\ y^{(p-1)}(0), ..., y'(0), y(0) \text{ donnés} \end{cases}$$

et les méthodes numériques que nous allons décrire pourront leur être appliquées.

4.1.1 Quelques soucis d'ordre théorique

Avant d'aborder l'étude numérique de (4.1), il est bon d'avoir à l'esprit quelques idées simples sur l'existence et le comportement des solutions de (4.1) :

Exemples linéaires :

$$\begin{cases} y' = ay \\ y(0) = y_0 \end{cases}, \; a \in \mathbb{R}$$

La solution est bien sûr $y(t) = y_0 e^{at}$. Bien qu'élémentaire, cet exemple nous permettra d'analyser quelques propriétés importantes des algorithmes étudiés plus loin.
Plus généralement, si a dépend de t, l'unique solution est donnée par

$$y(t) = y_0 \exp\left(\int_0^t a(\sigma)\, d\sigma\right)$$

dès que a est une fonction continue de t.
Considérons

$$\begin{cases} y' = Ay \\ y(0) = y_0 \in \mathbb{R}^N \end{cases} \tag{4.4}$$

où $y : [0, +\infty[\to \mathbb{R}^N$ et A est une matrice d'ordre N. Si A est diagonale, le système se découple en N équations scalaires indépendantes. On a alors

$$y_i(t) = y_0^i e^{\lambda_i t} \quad \text{si} \quad A = \begin{bmatrix} \ddots & & 0 \\ & \lambda_i & \\ 0 & & \ddots \end{bmatrix}.$$

Plus généralement, pour une matrice A quelconque, la solution est encore donnée par

$$y(t) = e^{tA} y_0$$

où e^{tA} est la matrice définie par :

$$e^{tA} = \sum_{n=0}^{\infty} \frac{t^n A^n}{n!}.$$

Lorsque la matrice A dépend raisonnablement du temps, on montre encore que (4.4) a une solution unique existant pour tout $t > 0$. Ces deux propriétés peuvent malheureusement aisément disparaître lorsque la fonction f est non linéaire.

Exemples non linéaires

$$\begin{cases} y' = y^2 \\ y(0) = y_0 \end{cases} \tag{4.5}$$

$$\begin{cases} y' = y^{\frac{1}{3}} \\ y(0) = y_0 \end{cases} \tag{4.6}$$

L'équation (4.5) s'intègre explicitement puisque, sur un intervalle où y ne s'annule pas :

$$y' = y^2 \Longleftrightarrow y' y^{-2} = 1 \Longleftrightarrow \frac{d}{dt}\left(-\frac{1}{y}\right) = 1 \Longleftrightarrow \frac{1}{y_0} - \frac{1}{y(t)} = t \Longleftrightarrow y(t) = \frac{y_0}{1 - t y_0}.$$

Ainsi, pour $y_0 > 0$, la solution explose pour $t^* = \frac{1}{y_0}$: il n'y a donc pas existence globale pour tout $t > 0$.
L'équation (4.6) s'intègre de façon identique : on obtient

$$y' y^{-\frac{1}{3}} = 1 \Longleftrightarrow \frac{d}{dt}\left(\frac{3}{2} y^{\frac{2}{3}}\right) = 1 \Longleftrightarrow \frac{3}{2} y^{\frac{2}{3}}(t) = \frac{3}{2} y_0^{\frac{2}{3}} + t.$$

En particulier, si $y_0 = 0$, on obtient comme solution

$$y(t) = \left(\frac{2}{3} t\right)^{\frac{3}{2}}.$$

Seulement, on vérifie que $y(t) = 0$ est aussi solution. Donc, il n'y a pas unicité des solutions de (4.6). En fait il y en a une infinité puisque pour tout $a > 0$

$$y_a(t) = \begin{cases} \left(\frac{2}{3}(t-a)\right)^{\frac{3}{2}} & \text{si } \ t \geq a \\ 0 & \text{si } \ 0 \leq t \leq a \end{cases}$$

est aussi une solution. Enonçons sans démonstration un théorème classique relativement général assurant existence et unicité locale.

Théorème 4.1 (Cauchy-Lipschitz) *i) On suppose que f est continue de $[0,\alpha] \times \mathbb{R}^N \to \mathbb{R}^N$ avec $\alpha > 0$. Alors, pour tout y_0, il existe au moins une solution continument dérivable sur $[0,T]$ de :*

$$\begin{cases} y'(t) = f(t, y(t)) & 0 \leq t \leq T \\ y(0) = y_0 \end{cases} \tag{4.7}$$

pour un certain $T \in]0,\alpha]$.
ii) On suppose de plus que f est localement lipschitzienne par rapport à la variable y, c'est-à-dire :

$$\|f(t,y_1) - f(t,y_2)\| \leq L(M)\|y_1 - y_2\| \ \text{ pour tout } t \in [0,T], \tag{4.8}$$

$$\text{pour tout } \ \ y_1,\ y_2 \text{ avec } \|y_1\| \leq M,\ \|y_2\| \leq M$$

où L est une fonction croissante de $[0,\infty[$ dans $[0,\infty[$ et $\|.\|$ une norme sur $\mathbb{R}^N$.
Alors, la solution de (4.7) est unique.

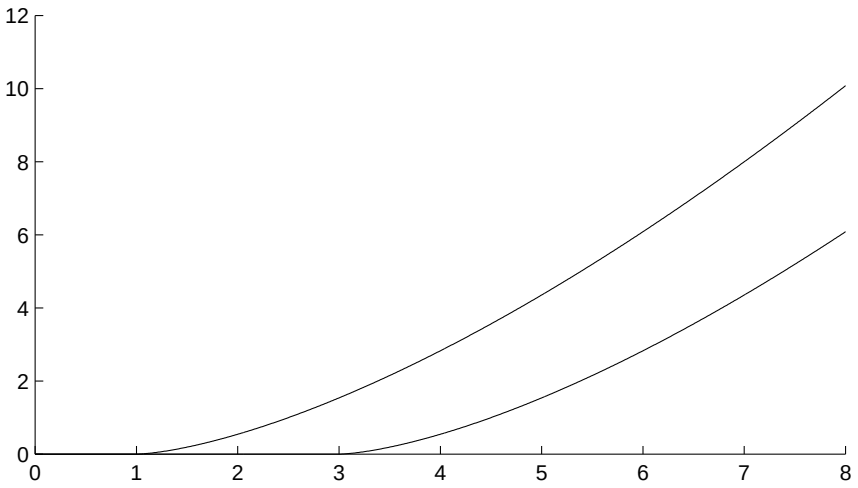

FIGURE 4.1 – Deux solutions distinctes de l'équation différentielle (4.6)

Remarque 4.1 Le point i) justifie l'existence de solutions locales pour les problèmes (4.5),(4.6) (voir aussi l'exercice 4.2). En revanche, on voit qu'elles ne sont pas nécessairement globales.

Les exemples (4.4) et (4.5) entrent dans le cadre (ii). En effet :

$$\left|y_1^2 - y_2^2\right| = |y_1 + y_2|\,|y_1 - y_2| \leq 2M\,|y_1 - y_2| \text{ si } |y_1|\,,\ |y_2| \leq M.$$

Par contre $y \to y^{\frac{1}{3}}$ n'est pas localement lipschitzienne.
On peut en fait montrer que les conclusions de ii) restent valables lorsque (4.8) est remplacé par l'hypothèse de monotonie (plus faible que (4.8)) :

$$\langle f(t, y_1) - f(t, y_2), y_1 - y_2\rangle \leq L(M)\,\|y_1 - y_2\|^2 \tag{4.9}$$

où $\langle .,. \rangle$ est le produit scalaire usuel sur $\mathbb{R}^N$ et $\|.\|$ la norme associée.
Ainsi, dans l'exercice 4.2, f vérifie (4.9) avec $L(M) \equiv 0$ (alors qu'elle ne vérifie pas (4.8).

Les hypothèses de monotonie assurent aussi l'existence globale :

Théorème 4.2 *On suppose que f est continue sur $[0, +\infty[\times\mathbb{R}^N$ et que*

$$\langle f(t, y_1) - f(t, y_2), y_1 - y_2\rangle \leq L\,\|y_1 - y_2\|^2 \ \forall y_1, y_2 \in \mathbb{R}^N,\ \forall t > 0. \tag{4.10}$$

Alors le problème (4.7) admet une solution et une seule sur $[0, \infty[$.

Remarque 4.2 Les hypothèses (4.10) recouvrent le cas où f est globalement lipschitzienne en y, soit

$$\|f(t, y_1) - f(t, y_2)\| \leq L\,\|y_1 - y_2\|\,. \tag{4.11}$$

C'est en particulier le cas si $f(t, y) = A(t)\,y$ où $A(t)$ est une matrice à coefficients continus et bornés sur $[0, +\infty[$.

Nous renvoyons à la littérature pour une démonstration de ces résultats (par exemple : "Analyse Numérique des équations différentielles", par M. Crouzeix et A.L. Mignot, Dunod).

4.1.2 Régularité de la solution

Par définition, une solution de (4.7) est une fonction dérivable sur $[0,T[$ satisfaisant

$$y'(t) = f(t, y(t)) \quad \forall t \in [0,T],\ y(0) = y_0. \tag{4.12}$$

Si f est continue, par composition, y' est continue. Donc y est une fonction continument dérivable : y est de classe C^1.
Supposons $N = 1$ et que f soit non seulement continue, mais aussi de classe C^1, i.e. $\frac{\partial f}{\partial t}(t,y)$ et $\frac{\partial f}{\partial y}(t,y)$ existent et sont continues sur $[0,T]$. Alors, d'après (4.12), $y'(t)$ est lui-même dérivable et

$$y''(t) = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t))\, y'(t).$$

Ceci montre que y'' est continue : y est de classe C^2. Le raisonnement peut être reconduit : y sera de classe C^{p+1} si f est de classe C^p. Par ailleurs, si $\frac{\partial f}{\partial y}$ est continue, d'après le théorème des accroissements finis, on a, pour $|y_1|,\ |y_2| \leq M,\ t \in [0,T]$

$$|f(t,y_1) - f(t,y_2)| \leq L(M)\, \|y_1 - y_2\|$$

où $L(M) = \max_{|y|\leq M,\ 0\leq t\leq T} \left|\frac{\partial f}{\partial t}(t,y)\right|$ et donc f est localement lipschitzienne.
L'hypothèse f est de classe C^1 nous assure existence, unicité, et régularité locale de la solution.
Ces remarques ne sont pas particulières à $N = 1$ mais restent valables pour les systèmes : seules les écritures sont plus lourdes.

4.2 Méthodes de résolution à un pas

Nous allons commencer par décrire une méthode très simple mais fondamentale : nous mettrons à profit sa simplicité pour en analyser son erreur de discrétisation, sa consistance, sa convergence et sa stabilité, autant de notions clefs dans l'analyse numérique des équations différentielles.

4.2.1 La méthode d'Euler

Soit donc à résoudre numériquement

$$\begin{cases} y'(t) = f(t, y(t)) & 0 \leq t \leq T \\ y(0) = y_0 \end{cases} \tag{4.13}$$

où $y : [0,+\infty[\to \mathbb{R}^N$ est la fonction cherchée, $f : [0,+\infty[\times \mathbb{R}^N \to \mathbb{R}^N,\ y_0 \in \mathbb{R}^N$. Nous supposerons que (4.13) admet une solution sur $[0,T]$. Nous noterons $M_0 = \max_{0\leq t\leq T} \|y(t)\|$ où $\|.\|$ est une norme sur $\mathbb{R}^N$ et nous supposerons que f est de classe C^1 sur $[0,T] \times \mathbb{R}^N$. Ainsi, d'après les remarques précédentes, nous aurons

$$\forall y_1, y_2 \text{ avec } \|y_1\|, \|y_2\| \leq M \qquad \|f(t,y_1) - f(t,y_2)\| \leq L(M)\, \|y_1 - y_2\| \tag{4.14}$$

$$\forall t \in [0,T] \quad \left\|y'(t)\right\| \leq M_1,\ \left\|y''(t)\right\| \leq M_2.$$

Nous allons discrétiser (4.13) en introduisant une subdivision

$$0 = t_0 < t_1 < t_2 < ... < t_n = T.$$

On notera $h_n = t_{n+1} - t_n,\ n = 0, ..., N-1$ et $h = \max_{0 \leq n \leq N} h_n$. Alors (4.13) implique

$$\begin{cases} \text{pour } n = 0, ..., N-1 & y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \\ y(0) = y^0 \end{cases} . \tag{4.15}$$

Les méthodes numériques utilisées pour résoudre (4.13) diffèrent par le choix de l'évaluation numérique des intégrales $\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt$.
La méthode d'Euler classique correspond à une méthode des rectangles à gauche, soit :

$$\begin{cases} \text{pour } n = 0, ..., N-1 & y_{n+1} = y_n + h_n f(t_n, y_n). \\ y_0 = y_0^h \text{ (valeur aprochée de } y^0). \end{cases} \tag{4.16}$$

Il est nécessaire d'analyser dans quelle mesure la valeur calculée approche suffisamment la

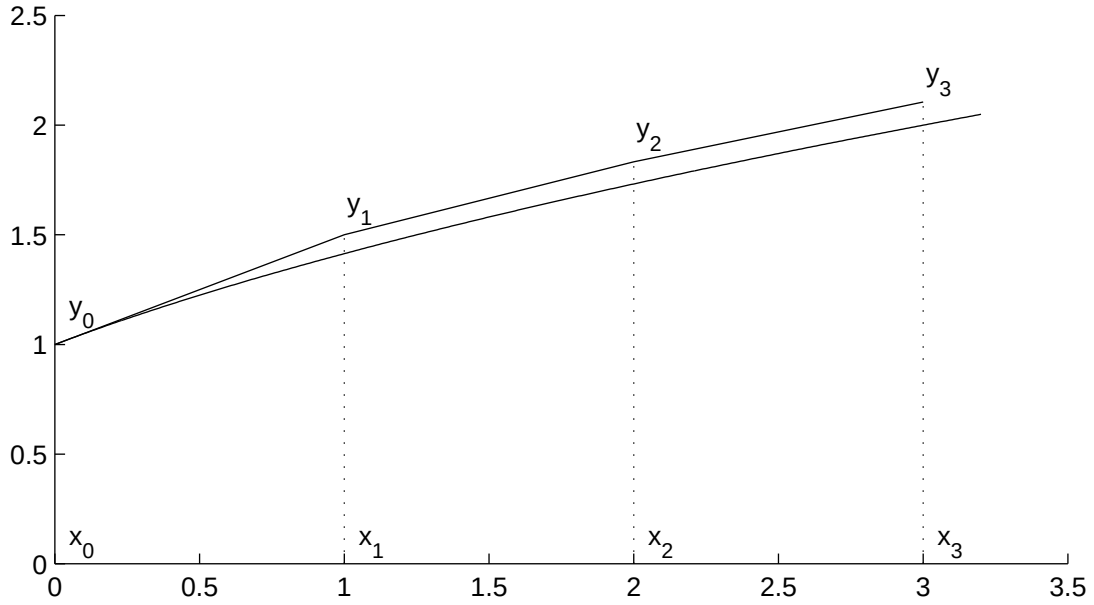

FIGURE 4.2 – La méthode d'Euler pour l'équation y'=1/(2y) avec un pas constant $h_n = 1$: $y_{n+1} = y_n + 1/(2y_n)$

valeur exacte $y(t_n)$ et donc d'évaluer l'erreur de discrétisation

$$e_n = y(t_n) - y_n.$$

Définition 4.1 *On dira que la méthode est convergente si*

$$\max_{0 \leq n \leq N} \|e_n\| \text{ tend vers 0 lorsque } h \text{ tend vers 0 et } y_0^h \text{ tend vers } y^0.$$

Remarque 4.3 si la méthode est convergente, en choisissant h assez petit, et y_0^h assez voisin de y^0, on obtient une bonne approximation de $y(t_n)$, $n = 0, ..., N-1$ à l'aide du schéma (4.16). Il semblerait plus naturel de poser directement $y_0 = y^0$ dans (4.16). Cependant, dans la pratique, si y^0 est réel, il ne pourra pas toujours être pris en compte de façon exacte à cause des arrondis. Une analyse correcte nécessite donc de supposer $y^0 \neq y_0^h$.

De plus, comme dans tout calcul numérique, il se pose un problème de stabilité, il est nécessaire de connaître les conséquences d'une petite variation de y_0^h ainsi que celles d'erreurs (inévitables dans la pratique) sur le calcul de $f(t_n, y_n)$.

Définition 4.2 *Nous dirons que la méthode (4.16) est stable s'il existe une constante K telle que*

$$\max_n \|y_n - z_n\| \leq K \left[\|y_0 - z_0\| + \sum_{n=0}^{N-1} \|\varepsilon_n\| \right] \tag{4.17}$$

pour tout z_n solution de

$$z_{n+1} = z_n + h_n f(t_n, z_n) + \varepsilon_n, \ n = 0, ..., N-1.$$

Cette notion de stabilité implique que de petites perturbations sur les données et sur tous les calculs intermédiaires n'entraînent que de "petites" perturbations du résultat ce qui est absolument nécessaire pour qu'un schéma numérique soit réaliste. Cependant, cette remarque n'est valable de façon pratique que si K est de taille raisonnable (voir exemples plus loin).
Enfin, on dégage aussi une notion de consistance d'un schéma numérique : l'erreur de consistance représente l'erreur qu'on fait au n-ième pas en remplaçant l'équation différentielle par l'équation discrétisée, soit ici :

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - h_n f(t_n, y(t_n)). \tag{4.18}$$

Définition 4.3 *On dit que la méthode est consistante si*

$$\lim_{h \to 0} \sum_{n=0}^{N-1} \|\varepsilon_n\| = 0.$$

Remarque 4.4 On montre que, pour les méthodes à un pas, consistance et stabilité impliquent convergence. Nous allons le constater ici pour la méthode d'Euler.

4.2.1.1 Estimation de l'erreur de discrétisation

$$e_{n+1} = y(t_{n+1}) - y_{n+1} = y(t_{n+1}) - y_n - h_n f(t_n, y_n)$$

soit d'après (4.18)

$$e_{n+1} = e_n + h_n (f(t_n, y(t_n)) - f(t_n, y_n)) + \varepsilon_n. \tag{4.19}$$

Pour des raisons de simplicité, nous supposerons ici que la constante de Lipschitz dans (4.8) est uniforme, c'est-à-dire

$$L(M) \leq L \qquad \forall M > 0 \tag{4.20}$$

De (4.19), (4.8), (4.20) on déduit

$$\|e_{n+1}\| \leq (1 + Lh_n) \|e_n\| + \|\varepsilon_n\| . \tag{4.21}$$

Afin de faciliter la récurrence dans (4.21), on utilise l'inégalité $1 + x \leq e^x$ pour obtenir

$$\|e_{n+1}\| \leq e^{Lh_n} \|e_n\| + \|\varepsilon_n\| .$$

Par récurrence, on obtient ainsi

$$\|e_n\| \leq e^{Lt_n} \|e_0\| + \sum_{i=0}^{n-1} e^{L(t_n - t_{i+1})} \|\varepsilon_i\| . \tag{4.22}$$

Remarque 4.5 Le calcul que nous venons de faire prouve en fait la stabilité de la méthode ; en effet, posant

$$z_{n+1} = z_n + h_n f(t_n, z_n) + \varepsilon_n \tag{4.23}$$

au lieu de (4.18) et en posant $\widehat{e}_n = z_n - y_n$ on aurait de même

$$\|\widehat{e}_n\| \leq e^{Lt_n} \|e_0\| + \sum_{i=0}^{n-1} e^{L(t_n - t_{i+1})} \|\varepsilon_i\| . \tag{4.24}$$

et donc

$$\max_{0 \leq n \leq 1} \|\widehat{e}_n\| \leq K \left[\|\widehat{e}_0\| + \sum_{i=0}^{n-1} \|\varepsilon_i\| \right] \tag{4.25}$$

avec $K = e^{LT}$.

Si nous revenons à (4.22), nous voyons que, lorsque h tend vers 0, $\|e_0\| = \left\|y^0 - y_0^h\right\|$ tend vers 0. Reste à montrer que le dernier terme tend vers 0, c'est-à-dire montrer la consistance de la méthode, puisque ce dernier terme est majoré par $e^{LT} \sum_{i=0}^{N-1} \|\varepsilon_i\|$.
Or

$$\begin{aligned} \varepsilon_n &= y(t_{n+1}) - y(t_n) - h_n f(t_n, y(t_n)) = \int_{t_n}^{t_{n+1}} \left(y'(t) - y'(t_n) \right) dt \\ &= \int_{t_n}^{t_{n+1}} \left[\int_{t_n}^{t} y''(\sigma)\, d\sigma \right] = \int_{t_n}^{t_{n+1}} (t_{n+1} - \sigma)\, y''(\sigma)\, d\sigma. \end{aligned}$$

On retiendra de ceci que

$$\|\varepsilon_n\| \leq h \int_{t_n}^{t_{n+1}} \left\| y''(\sigma) \right\| d\sigma \leq hM_2 (t_{n+1} - t_n)$$

$$\sum_{i=0}^{n-1} \|\varepsilon_i\| \leq hM_2 t_n. \tag{4.26}$$

Ainsi, ces estimations jointes à (4.22) donnent

$$\|e_n\| \leq e^{Lt_n} [\|e_0\| + hM_2 t_n]$$

et

$$\max_{0 \leq n \leq N} \|e_n\| \leq e^{LT} [\|e_0\| + hM_2 T] \tag{4.27}$$

ce qui prouve que la méthode d'Euler est convergente.

4.2.1.2 Influence des erreurs d'arrondis

Si on veut calculer $y(T)$ avec une précision donnée ε, d'après (4.27), il faut bien sûr d'abord choisir $\|e_0\|$ assez petit : supposons qu'on puisse le choisir nul. En théorie, on peut alors choisir h assez petit pour que $hM_2T < \varepsilon$. Cependant, dans la pratique, il est illusoire de vouloir choisir h "trop" petit, car alors les erreurs d'arrondis prennent le relais de l'erreur de discrétisation : en effet, le calcul

$$y_{n+1} := y_n + h_n f(t_n, y_n)$$

donne lieu à une erreur μ_n sur le calcul de $f(t_n, y_n)$ et à une erreur d'arrondi ρ_n sur le résultat final : l'algorithme du calculateur est donc

$$z_{n+1} = z_n + h_n (f(t_n, z_n) + \mu_n) + \rho_n.$$

D'après les formules (4.23), (4.24), (4.25), on a

$$\|y_N - z_N\| \leq e^{LT} \left(\|y_0 - z_0\| + \sum_{i=0}^{N-1} (h_n \|\mu_n\| + \|\rho_n\|) \right).$$

Supposons $\|\mu_n\| \leq \mu$ et $\|\rho_n\| \leq \rho$. La somme de l'erreur de discrétisation $\|y(T) - y_N\|$ et des erreurs d'arrondis est donc majorée par (remarquer $\sum_{n=0}^{n-1} \|\rho_n\| \leq N\rho = \frac{T\rho}{h}$) :

$$E_N = e^{LT} \left(hM_2T + \|y_0 - z_0\| + T\mu + \frac{T}{h}\rho \right). \tag{4.28}$$

Ceci montre que les erreurs d'arrondis tendent en général vers l'infini lorsque h tend vers 0 ... ce qui est bien normal puisqu'on accumule alors le nombre d'opérations. On peut essayer de voir quel serait le pas "optimal" qui rendrait E_N minimum. Puisque E_N est de la forme

$$\begin{aligned} E_N &= A + Bh + \rho\frac{C}{h} \\ \text{avec } A &= e^{LT}(\|y_0 - z_0\| + T\mu) \\ B &= M_2 T e^{LT} \\ C &= T e^{LT}, \end{aligned}$$

le minimum est atteint pour $h = h^* = \sqrt{\rho\frac{C}{B}}$: c'est-à-dire que, autour de h^* les erreurs d'arrondis s'équilibrent avec celles dues à la méthode. Un bon choix nécessitera bien sûr $h \geq h^*$: il est inutile d'augmenter le coût de calcul pour rien.

Remarque 4.6 Les calculs ci-dessus sont faits de façon "pessimiste" : souvent les erreurs d'arrondis se compensent dans une certaine mesure (mais c'est aléatoire).

Il est intéressant de noter dans l'expression de E_N comment les erreurs d'arrondi sur la donnée initiale $\|y_0 - z_0\|$ se "propagent" : leur impact sur l'erreur finale est majoré par $e^{LT}\|y_0 - z_0\|$ ce qui est raisonnable ... si e^{LT} n'est pas trop grand : nous allons voir sur des exemples ce que ceci signifie numériquement :

Exemple 1 :

$$\begin{cases} y'(t) = 3y(t) - 3t & t \in [0, 6] \\ y(0) = \frac{1}{3} \end{cases} \tag{4.29}$$

La solution est $y(t) = \frac{1}{3} + t$. Ce problème est donc tout à fait inoffensif. Cependant, ici $L = 3$ donc pour $T = 6$,

$$e^{LT} = e^{18} \simeq 0.6510^8.$$

Ainsi, si on considère la solution $z(t)$du même problème avec la donnée initiale $z(0) = \frac{1}{3} + \varepsilon (= 0.33333333$ par exemple$)$; par différence

$$\begin{cases} (z-y)' = 3(z-y) \\ (z-y)(0) = \varepsilon \end{cases} \quad \text{et donc } (z-y)(t) = \varepsilon e^{3t}.$$

Ainsi $z(T) = y(T) + \varepsilon e^{18} \simeq y(T) + 0.65\varepsilon 10^8$. En conséquence, si on travaille avec un calculateur travaillant avec une présence relative de 10^{-8}, il sera pratiquement impossible d'approcher convenablement $y(6)$ ... et ceci indépendamment du choix de la méthode numérique. On dit que le problème ci-dessus est mal conditionné ou même ici, numériquement mal posé.

Exemple 2 :

$$\begin{cases} y'(t) = -150\, y(t) + 50 \\ y(0) = \frac{1}{3} \end{cases} . \tag{4.30}$$

La solution est $y(t) \equiv \frac{1}{3}$. La méthode d'Euler conduit à

$$y_{n+1} = y_n + h_n(-150 y_n + 50).$$

Soit encore

$$y_{n+1} - \frac{1}{3} = (1 - 150 h_n)\left(y_n - \frac{1}{3}\right).$$

Si $h_n = h = \frac{1}{50}$, on a donc

$$y_n - \frac{1}{3} = (1-150h)^n \left(y_0 - \frac{1}{3}\right) = (-2)^n \left(y_0 - \frac{1}{3}\right). \tag{4.31}$$

En particulier,

$$y_{50} - y(1) = (-2)^{50}\left(y_0 - \frac{1}{3}\right)!$$

Sachant que $2^{50} \simeq 10^{15}$, ceci montre que le pas choisi est trop grand.
Ici, la méthode d'Euler n'est pas un algorithme numériquement stable : bien qu'elle satisfasse à la condition de stabilité théorique (4.17), le coefficient K est trop grand. Cependant, on peut montrer que ce problème est bien conditionné au sens qu'une petite variation de y_0 induit une petite variation de $y(t)$ puisque, grâce au signe $-$ devant $150y(t)$, on montre aisément :

$$|y(t) - \widehat{y}(t)| \leq |y_0 - \widehat{y}_0|.$$

La relation (4.31) met en évidence un autre phénomène : lorsque n grandit, $y_n - \frac{1}{3}$ devient en module de plus en plus grand et est alternativement positif et négatif : au bout d'un temps très court, les calculs ne sont plus significatifs. Analysons ceci de façon un peu plus systématique.

Exemple 3 :

$$\begin{cases} y'(t) = -\lambda y(t) & \lambda > 0 \\ y(0) = y_0 \end{cases} . \tag{4.32}$$

Ce problème continu est très stable par rapport à y_0 puisque $y(t) = y_0 e^{-\lambda t}$ et donc

$$|y(t) - \widehat{y}(t)| \leq e^{-\lambda t} |y_0 - \widehat{y}_0| .$$

La méthode d'Euler appliquée à (4.32) avec pas constant donne

$$y_{n+1} = y_n - \lambda h y_n = (1 - \lambda h) y_n$$

et donc

$$y_n = (1 - \lambda h)^n y_0.$$

Le pas h étant fixé, on voit que, bien que la solution exacte soit bornée, la solution calculée y_n aura un module de plus en plus grand si $|1 - \lambda h| > 1$, d'où un phénomène d'instabilité. La condition de stabilité s'écrit donc

$$\lambda h < 2 \tag{4.33}$$

Ainsi, il sera nécessaire de prendre h d'autant plus petit que λ est grand. Cette restriction apparaît également très souvent dans les systèmes différentiels provenant de la discrétisation d'équations aux dérivées partielles de type parabolique (voir la condition CFL du chapitre précédent).

4.2.2 Méthode d'Euler implicite

Afin de remédier au problème d'instabilité soulevé ci-dessous, on a souvent recours à une méthode de type implicite comme la suivante

$$\begin{cases} y_{n+1} = y_n + h_n f(t_{n+1}, y_{n+1}) \\ y_0 = y^h \end{cases} \tag{4.34}$$

qui provient de l'intégration de $\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt$ par une méthode des rectangles à droite. On voit que la relation (4.34) définit y_{n+1} de façon implicite et qu'il faut donc résoudre un problème pour calculer y_{n+1}. Le coût est évidemment plus lourd que dans la méthode d'Euler explicite vue précédemment, mais en contrepartie la stabilité est grandement améliorée comme on va le voir en reprenant l'exemple (4.32). On a donc, avec un pas constant :

$$y_{n+1} = y_n - \lambda h y_{n+1}$$

soit

$$y_{n+1} = \frac{y_n}{1 + \lambda h},\ y_n = \frac{y_0}{(1 + \lambda h)^n}.$$

En particulier, on a toujours (quel que soit le choix du pas h)

$$|y_n| \leq |y_0| .$$

Par ailleurs, on montre que cette méthode converge avec la même vitesse que la précédente.

4.3 Résolution d'EDO en utilisant Matlab

Matlab dispose de plusieurs solveurs d'équations différentielles :
- $ode23$ et $ode45$ qui codent des méthodes classiques de Runge-Kutta (cf section 5.1 dans le chapitre 5)
- $ode113$ qui utilise une méthode d'Adams-Bashforth (cf section 5.3.1 dans le chapitre 5)
- $ode15s, ode23s$ qui sont à utiliser lorsque l'équation différentielle à résoudre est particulièrement *raide* (stiff en anglais) ce qui peut arriver notamment en cinétique chimique.

Lorsqu'il n'y a pas lieu de croire que l'équation est problématique, on peut commencer par $ode45$ ou $ode23$. Ils s'utilisent comme ceci :

* L'équation différentielle doit être ramenée (comme à la section 4.1) à une équation, eventuellement vectorielle, mais du premier ordre en temps ; par exemple l'équation $x^{''} + 2x^{'} - tx = 0$, s'écrit

$$y^{'} = \begin{pmatrix} 0 & 1 \\ t & -2 \end{pmatrix} y \quad \text{où on a posé} \quad y(t) = \begin{pmatrix} x(t) \\ x^{'}(t) \end{pmatrix}$$

* Le système est codé sous forme d'un fichier qui étant donné y et t renvoie $y^{'}(t)$; créons ici le fichier *essai.m* suivant :

$$function\ dy = essai(t, y);$$
$$dy = [y(2);\ t * y(1) - 2 * y(2)];$$

Dans le cas où l'équation est vectorielle, la dérivée doit obligatoirement être un vecteur colonne et non un vecteur ligne. Dans le cas des équations à coefficients constants, il est nécessaire de taper la ligne *function dy = essai(t,y)* même si dy ne dépend pas du temps t.

* On peut ensuite utiliser un solveur Matlab avec la commande :

$$[t, y] = ode23('essai', [tinitial, tfinal], y0)$$

où $tinitial$ et $tfinal$ sont les instants entre lesquels on cherche à approcher la solution et où $y0$ est la condition initiale.
Les algorithmes sont à pas adaptatifs le vecteur t contient les instants pour lesquels une valeur approchée de la solution y a été calculée ; si l'on souhaite connaitre la solution à des instants bien particulier, par exemple pour un pas de temps régulier, il suffit de remplacer le vecteur $[tinitial, tfinal]$ par le vecteur temps souhaité.

4.4 Exercices du chapitre 4

Exercice 4.1 Résoudre le système (4.4) quand la matrice A est diagonalisable.

Exercice 4.2 Vérifier que les problèmes

$$y' = -y^2 \quad y(0) = y_0 \tag{4.35}$$

$$y' = -y^{\frac{1}{3}} \quad y(0) = y_0 \tag{4.36}$$

ont une solution unique sur $[0, \infty[$.

Exercice 4.3 Soit $y'(t) = Ay(t)$ où A est une matrice carrée $N \times N$ diagonalisable de valeurs propres $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$.
Montrer que si $\lambda_N h < 2$, la méthode d'Euler est stable au sens que y_n reste borné pour n grand.

Exercice 4.4 Pour résoudre numériquement $y' = -\lambda y$, on décide d'approcher $y'(t_n)$ par les différences finies centrées

$$y'(t_n) \simeq \frac{y(t_{n+1}) - y(t_{n-1})}{2h},\ h = t_{n+1} - t_{n-1},\ \forall n.$$

Ceci conduit au schéma

$$y_{n+1} = y_{n-1} - 2\lambda h y_n.$$

Montrer que ce schéma est toujours instable.

Exercice 4.5 On considère l'équation différentielle

$$\begin{cases} y'(t) = e^{-ty(t)} \\ y(0) = 0\ . \end{cases}$$

Montrer l'existence et l'unicité locale, puis globale sur $\mathbb{R}$ de la solution. En déduire que celle-ci est impaire. Montrer qu'elle admet une limite quand $t \to +\infty$.

4.5 TD4 : Résolution numérique d'équations différentielles

Exercice 1 Stabilité théorique et stabilité numérique

Résoudre théoriquement l'équation différentielle suivante :

$$\begin{cases} y^{'}(t) = 3y(t) - 3t \\ y(0) = \frac{1}{3} \end{cases}$$

Ce problème risque-t-il d'être instable ? Résoudre ce problème à l'aide des solveurs *ode*23 et *ode*45 entre 0 et *tmax* pour *tmax* compris entre 5 et 20. Que constate-t-on ? Comparer les 3 solutions en les traçant sur un même graphique. Expliquer ce que vous observez.

Exercice 2 :
Soit l'équation différentielle suivante :

$$\begin{cases} y^{'}(t) = \exp\left(y(t)\right) \\ y(0) = y_0 = 1 \end{cases}$$

1. Résoudre de façon approchée cette équation à l'aide d'un solveur Matlab. Qu'observe-t-on ?

2. Retrouver cette propriété en résolvant théoriquement l'équation différentielle.

On se propose de mettre en place les deux méthodes d'Euler à pas fixe pour proposer une valeur approchée du temps maximun d'existence.

3. Méthode d'Euler explicite
Programmer la méthode d'Euler explicite à pas constant. Quel test d'arrêt proposez-vous ? Quelle est, en fonction du pas de temps h, la précision dont on dispose sur le temps maximum d'existence ?

4. Méthode d'Euler implicite
Programmer maintenant la méthode d'Euler implicite à pas constant. Vous effectuerez la résolution de l'équation non linéaire, à chaque pas, grâce à la fonction Matlab *fzero*.

4.6 Aide-mémoire : équations différentielles linéaires

Il y a quelques équations différentielles qu'on sait résoudre explicitement. L'exemple le plus notable est celui des équations linéaires à coefficients constants. Remarquons pour commencer que le Théorème de Cauchy-Lipschitz 4.1 a une jolie application dans ce cas :

Théorème 4.3 *Soit $a_0, a_1, \ldots, a_p$ des réels avec $a_p \neq 0$. Alors l'ensemble des solutions de l'équation différentielle*

$$a_p y^{(p)} + \ldots + a_1 y' + a_0 y = 0 \tag{4.37}$$

est un espace vectoriel de dimension p.

Le fait que ce soit un espace vectoriel E est évident ici et est dû au caractère linéaire de l'équation. Pourquoi est-il de dimension p ? Considérons l'application

$$\Phi : \begin{array}{ccc} E & \longrightarrow & \mathbb{R}^p \\ y & \mapsto & (y(0), y'(0), \ldots, y^{(p-1)}(0)) \end{array}$$

Le Théorème de Cauchy-Lipschitz permet de voir que cette application Φ est bijective : en effet, pour tout élément $(b_1, b_2, \ldots, b_p)$ de $\mathbb{R}^p$, il existe une unique solution de l'équation différentielle (4.37) qui satisfait $(y(0), y'(0), \ldots, y^{(p-1)}(0)) = (b_1, b_2, \ldots, b_p)$. Autrement dit, tout élément de l'espace d'arrivée $\mathbb{R}^p$ a un unique antécédent.

Maintenant qu'on sait que E est de dimension p, pour le déterminer, il suffit d'exhiber une famille de p solutions qui soient linéairement indépendantes (et qui formeront alors une base de l'espace vectoriel E). A cet effet, on peut chercher des solutions sous la forme exponentielle : $y(x) = \exp(\lambda x)$. En substituant dans l'équation (4.37). Cela conduit alors à **l'équation caractéristique de l'équation (4.37)** :

$$a_p \lambda^p + \ldots + a_1 \lambda + a_0 = 0 \tag{4.38}$$

dont toute racine $\hat{\lambda}$ (et on sait qu'il yen a exactement p, quitte à travailler en nombres complexes) va fournir une solution de l'équation différentielle sous la forme $\hat{y}(x) = \exp(\hat{\lambda} x)$. Par exemple, si l'équation caractéristique (4.38) a p racines distinctes (ce qui est le cas le plus fréquent) $\lambda_1, \ldots, \lambda_p$ alors il est possible de vérifier que les p fonctions $y_k(x) = \exp(\lambda_k x)$ sont linéairement indépendantes (exercice !) et on a donc démontré que toutes les solutions de l'équation différentielle (4.37) peuvent s'écrire

$$y(x) = \sum_{k=1}^{p} c_k \exp(\lambda_k x)\,.$$

Dans le cas où l'équation caractéristique possède des racines multiples, il convient de rajouter des solutions sous la forme $x\exp(\lambda x), x^2 \exp(\lambda x)$... Notons que ce qu'on vient de présenter reste théorique et n'est pas nécessairement très utilisable pour un degré $p \geq 5$. En effet, la recherche de toutes les racines d'un polynôme de degré $p \geq 5$ n'est pas si facile (voir chapitre 8) et donc, les méthodes numériques présentées dans ce chapitre et dans le suivant s'avèreront bien souvent beaucoup plus efficaces.

Enfin, si l'équation a un second membre, c'est-à-dire si elle est de la forme

$$a_p y^{(p)} + \ldots + a_1 y' + a_0 y = g(x) \tag{4.39}$$

la technique consiste à résoudre d'abord l'équation homogène (4.37) comme on l'a décrit ci-dessus, puis à rechercher une solution particulière de l'équation complète par votre méthode favorite (essayer de deviner la forme de cette solution particulière, puis méthode des coefficients indéterminés ou bien méthode de variation de la constante). La solution générale de (4.39) s'obtient alors en ajoutant les deux.

Chapitre 5

Équations différentielles - Méthodes de Runge-Kutta et méthodes multi-pas

Nous envisageons maintenant d'autres méthodes de résolution de l'équation différentielle

$$\begin{cases} y'(t) = f\left(t, y(t)\right), \ t > 0 \\ y\left(0\right) = y_0 \qquad \text{donné} \end{cases} \tag{5.1}$$

plus rapides et plus efficaces que la méthode d'Euler. Nous commençons par exposer les méthodes de Runge-Kutta qui sont des méthodes *à un pas*, puis nous expliquons comment on peut adapter le pas de discrétisation. Nous décrivons les méthodes multi-pas. Nous présentons ensuite les avantages et inconvénients de chaque méthode. Nous terminons par évoquer le cas des *problèmes aux limites*.

5.1 Les méthodes de Runge-Kutta

Une méthode à un pas s'écrit de façon générale

$$y_{n+1} = y_n + h_n \phi\left(t_n, y_n, h_n\right). \tag{5.2}$$

Ainsi, y_{n+1} est calculé à partir de y_n par l'intermédiaire de la fonction ϕ. Cette méthode peut être explicite ou implicite. Ainsi, pour la méthode d'Euler explicite, on a

$$\phi\left(t, y, h\right) = f\left(t, y\right)$$

tandis que pour la méthode d'Euler implicite $\phi\left(t, y, h\right)$ est défini de façon implicite ; $\phi\left(t, y, h\right)$ est la solution de l'équation

$$Y = y_n + h_n f\left(t_{n+1}, Y\right).$$

Définition 5.1 *On dit que la méthode (5.2) est d'ordre p si l'erreur de consistance vérifie*

$$\sum_{n=0}^{N-1} \left\| y\left(t_{n+1}\right) - y\left(t_n\right) - h_n \phi\left(t_n, y\left(t_n\right), h_n\right) \right\| \leq K h^p \tag{5.3}$$

où $h = \max_{0 \leq n < N} h_n$ *et ce, pour toute solution* p *fois continument dérivable de* $y'(t) = f(t, y(t))$.

Exemple : Nous avons vu que la méthode d'Euler explicite est d'ordre 1 (cf. (4.26)). Il en est de même pour la méthode d'Euler implicite. Nous allons le montrer ici en considérant plus généralement :

θ-méthode :

$$y_{n+1} = y_n + h_n \left[\theta f(t_{n+1}, y_{n+1}) + (1-\theta) f(t_n, y_n)\right] \tag{5.4}$$

où $\theta \in [0,1]$. Pour $\theta = 0$, c'est la méthode d'Euler explicite ; pour $\theta = 1$ on retrouve la méthode d'Euler implicite (4.34).

Analysons l'erreur de consistance :

$$\varepsilon_n = y(t_{n+1}) - y(t_n) - h_n \left[\theta f(t_{n+1}, y(t_{n+1})) + (1-\theta) f(t_n, y_n)\right].$$

Pour simplifier, nous supposerons $N = 1$; les techniques et résultats sont identiques en dimension supérieure. D'après la formule de Taylor [1] :

$$y'(t_{n+1}) = y'(t_n) + h_n y''(t_n) + \frac{h_n^2}{2} y^{(3)}(c_n)$$

où $c_n \in [t_n, t_{n+1}]$. De même

$$y(t_{n+1}) = y(t_n) + h_n y'(t_n) + \frac{h_n^2}{2} y''(t_n) + \frac{h_n^3}{6} y^{(3)}(\widehat{c}_n).$$

On en déduit

$$\begin{aligned} \varepsilon_n &= h_n y'(t_n) + \frac{h_n^2}{2} y''(t_n) + \frac{h_n^3}{6} y^{(3)}(\widehat{c}_n) \\ &\quad - h_n \left[\theta \left(y'(t_n) + h_n y''(t_n) + \frac{h_n^2}{2} y^{(3)}(c_n)\right) + (1-\theta) y'(t_n)\right] \end{aligned}$$

soit

$$\varepsilon_n = h_n^2 y''(t_n) \left[\frac{1}{2} - \theta\right] + h_n^3 \left(\frac{1}{6} y^{(3)}(\widehat{c}_n) - \frac{\theta}{2} y^{(3)}(c_n)\right).$$

Si $\theta \neq \frac{1}{2}$ (par exemple $\theta = 0$ ou $\theta = 1$), l'erreur locale sera en h^2 et l'erreur totale de consistance sera en h et la méthode d'ordre 1. Par contre, si $\theta = \frac{1}{2}$, on obtient une méthode d'ordre 2. Ce choix est souvent appelé méthode de Crank-Nicholson.

Méthode de Crank-Nicholson :

$$y_{n+1} = y_n + h_n \left[\frac{1}{2} f(t_{n+1}, y_{n+1}) + \frac{1}{2} f(t_n, y_n)\right] \tag{5.5}$$

Analysons la stabilité asymptotique sur le cas particulier

$$y'(t) = -\lambda y(t).$$

On a alors

$$y_{n+1} = y_n + h\left[-\lambda\theta y_{n+1} - \lambda(1-\theta) y_n\right]$$

1. voir Section 1.7.1

$$y_{n+1} = \frac{1 - \lambda h (1-\theta)}{1 + \lambda h \theta} y_n$$

$$y_n = \left(\frac{1 - \lambda h (1-\theta)}{1 + \lambda h \theta} \right)^n y_0.$$

L'approximation y_n restera bornée pour n grand si

$$-1 < \frac{1 - \lambda h (1-\theta)}{1 + \lambda h \theta} < 1.$$

Ceci équivaut à

$$0 < \frac{\lambda h}{1 + \lambda h \theta} \leq 2 \text{ ou } 0 < \lambda h \leq 2 + \lambda h . 2\theta.$$

D'où la condition de stabilité asymptotique

$$\begin{cases} \cdot \text{ si } \theta \geq \frac{1}{2} : \text{méthode stable } \forall h > 0 \\ \cdot \text{ si } \theta < \frac{1}{2} : \text{méthode stable si } \lambda h < \frac{2}{1-2\theta} \end{cases} \tag{5.6}$$

Conclusion : La méthode de Crank-Nicholson est d'ordre 2 et stable (au sens ci-dessus) pour tout choix de h. Ceci explique qu'elle soit souvent retenue.

Dans la pratique, même une méthode d'ordre 2 se révèle assez souvent insuffisante car nécessitant trop de pas de temps pour atteindre une précision donnée. Il est alors nécessaire d'utiliser une méthode d'ordre supérieur : les plus courantes sont celles de Runge-Kutta qui reposent sur des méthodes d'intégration numérique d'ordre supérieur pour

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt.$$

Décrivons-en le principe : $t_{n,j} = t_n + h_n c_j,\ j = 1, ..., q$ étant des points de $[t_n, t_{n+1}]$ $(0 \leq c_j \leq 1)$, le remplacement de $f(t, y(t))$ par un polynôme d'interpolation aux points $t_{n,j}$ conduit à une formule d'intégration numérique du type

$$\int_{t_n}^{t_{n+1}} f(t, y(t))\, dt \simeq h_n \left(\sum_{j=1}^{q} b_j f(t_{n,j}, y(t_{n,j})) \right).$$

Les méthodes de Runge-Kutta consistent à remplacer ces évaluations approchées par des égalités soit

$$y_{n+1} = y_n + h_n \left[\sum_{j=1}^{q} b_j f(t_{n,j}, y_{n+j}) \right], \tag{5.7}$$

les valeurs $y_{n,j}$ étant elles-mêmes évaluées à l'aide de formules d'intégration numérique utilisant les mêmes points $t_{n,j}$:

$$\left\{ y_{n,j} = y_n + h_n \left[\sum_{j=1}^{q} a_{ij} f(t_{n,j}, y_{n,j}) \right], j = 1, ..., q \right\}. \tag{5.8}$$

Remarque 5.1 – Les formules (5.8) définissent les valeurs $y_{n,j}$ de façon explicite si la matrice $[a_{ij}]$ est strictement triangulaire inférieure, sinon elles sont obtenues de façon implicite.
– La méthode d'Euler et la θ-méthode sont des cas particuliers. Citons quelques méthodes d'ordre supérieur souvent utilisées.

Runge-Kutta d'ordre 2

$$\begin{cases} y_{n+1} = y_n + \frac{1}{2}\left(k_1^n + k_2^n\right) \\ k_1^n = h_n f\left(t_n, y_n\right) \\ k_2^n = h_n f\left(t_n + h_n, y_n + k_1^n\right) \end{cases} \tag{5.9}$$

ou

$$\begin{cases} y_{n+1} = y_n + k_2^n \\ k_1^n = h_n f\left(t_n, y_n\right) \\ k_2^n = h_n f\left(t_n + \frac{h_n}{2}, y_n + \frac{k_1^n}{2}\right) \end{cases} \tag{5.10}$$

et la fameuse méthode d'ordre 4 :

Runge-Kutta d'ordre 4

$$\begin{cases} y_{n+1} = y_n + \frac{1}{6}\left(k_1^n + 2k_2^n + 2k_3^n + k_4^n\right) \\ k_1^n = h_n f\left(t_n, y_n\right) \\ k_2^n = h_n f\left(t_n + \frac{h_n}{2}, y_n + \frac{k_1^n}{2}\right) \\ k_3^n = h_n f\left(t_n + \frac{h_n}{2}, y_n + \frac{k_2^n}{2}\right) \\ k_4^n = h_n f\left(t_{n+1}, y_n + k_3^n\right) \end{cases} . \tag{5.11}$$

On détermine l'ordre de ces méthodes à l'aide de la formule de Taylor. Considérons par exemple

$$\begin{cases} y_{n+1} = y_n + \left(ak_1^n + bk_2^n\right) \\ k_1^n = h_n f\left(t_n, y_n\right) \\ k_2^n = h_n f\left(t_n + \alpha h_n, y_n + \beta k_1^n\right) \end{cases} . \tag{5.12}$$

Il s'agit d'estimer

$$\varepsilon_{n+1} = y\left(t_{n+1}\right) - y\left(t_n\right) + h_n \phi\left(t_n, y\left(t_n\right), h_n\right)$$

où

$$\phi\left(t_n, y\left(t_n\right), h_n\right) = a\lambda + b\mu$$

avec

$$\begin{aligned} \lambda &= f\left(t_n, y\left(t_n\right)\right) \\ \mu &= f\left(t_n + \alpha h_n, y\left(t_n\right) + \beta h_n \lambda\right). \end{aligned}$$

Or

$$f\left(t_n + \alpha h_n, y\left(t_n\right) + \beta h_n \lambda\right) = f\left(t_n, y\left(t_n\right)\right) + \alpha h_n f_t\left(t_n, y\left(t_n\right)\right) + \beta h_n \lambda f_y\left(t_n, y\left(t_n\right)\right) + O\left(h_n^2\right).$$

$$y(t_{n+1}) - y(t_n) = h_n y'(t_n) + \frac{h_n^2}{2} y''(t_n) + O\left(h_n^3\right).$$

Puisque

$$y''(t_n) = f_t(t_n, y(t_n)) + f_y(t_n, y(t_n)).\lambda,$$

on en déduit

$$\varepsilon_{n+1} = h_n \lambda (1 - (a+b)) + h_n^2 \left(\frac{1}{2}(f_t + \lambda f_y) - b(\alpha f_t + \beta \lambda f_y) \right) + O\left(h_n^3\right).$$

L'erreur locale est d'ordre 3 (et la méthode d'ordre 2) si

$$\begin{cases} a+b=1 \\ \frac{1}{2} = \alpha b = b\beta \end{cases}.$$

C'est le cas dans les algorithmes (5.10) et (5.11).

Remarque 5.2 On démontre, sous des hypothèses de régularité sur f, que les méthodes de Runge-Kutta sont stables au sens de (4.17). Etant stables et consistantes, elles sont convergentes suivant un principe général déjà mentionné pour les méthodes à un pas.

5.2 Contrôle du pas

Dans la description des algorithmes précédents, nous avons supposé le pas h_n variable. Dans la pratique, il est naturel de chercher à utiliser un pas constant $h_n = h$ et on le fait souvent. Cependant, il est difficile d'évaluer la taille optimale du pas h : il ne doit pas être trop petit, sinon le coût de calcul est important et les erreurs d'arrondis s'accumulent. Il ne doit pas être trop grand pour que les calculs conservent une précision suffisante. Toute la difficulté consiste bien sûr à choisir le pas pour que la valeur $y(T)$ puisse être connue avec une précision préfixée.

Les algorithmes les plus performants de ce point de vue réalisent un contrôle du pas à chaque étape, dans le but

- de rester dans une marge de précision voulue
- mais aussi d'adapter le pas aux irrégularités éventuelles de la solution : sur des intervalles où la solution varie peu, on pourra "avancer à grands pas". Quand les variations deviennent plus raides, on diminue le pas.

Un contrôle "optimal" du pas dans un problème donné est toujours difficile à déterminer et, souvent, coûteux à mettre en oeuvre. Nous nous contenterons de donner ici quelques idées simples pouvant être utilisées en première approximation.

Etant donné une méthode à un pas de type (5.2), il s'agit à chaque étape de choisir un pas h tel que l'erreur $y(t_n + h) - y_n - h\phi(t_n, y_n, h)$ reste en deçà d'une certaine tolérance. Supposons avoir agi pour le mieux aux étapes précédentes : on ne peut plus alors agir sur y_n et (à moins d'espérer des erreurs de compensation qui sont de toute façon imprévisibles en général), on peut raisonner en supposant que y_n est connue de façon exacte et contrôler alors l'erreur locale de consistance :

$$\varepsilon_n(h) = y(t_n + h) - (y(t_n) + h_n \phi(t_n, y(t_n), h_n)).$$

(Un raisonnement fait de façon globale conduit en fait à la même conclusion : contrôler l'erreur locale de consistance). Ainsi, on pourra s'imposer que

$$|\varepsilon_n(h)| \leq \alpha h \tag{5.13}$$

où α est une certaine tolérance. L'erreur globale de discrétisation sera alors majorée par un facteur de α (cf. (4.22)) (reste bien sûr l'évaluation de ce facteur de type e^{LT}, difficulté toujours présente).
Tout le problème est d'évaluer l'erreur ε_n afin d'en déduire une valeur optimale de h satisfaisant à (5.13). Les diverses méthodes de contrôle du pas diffèrent essentiellement sur la technique d'évaluation de ε_n.
L'une d'elles consiste, étant donné un pas h provenant par exemple de l'étape précédente à calculer 2 valeurs approchées de $y(t_n+h)$, l'une $\overline{y}(t_n+h)$ à l'aide d'une itération de pas h, l'autre $\widehat{y}(t_n+h)$ avec deux itérations de pas $\frac{h}{2}$: l'erreur ε_n est alors évaluée à l'aide d'un procédé d'extrapolation à la limite de Richardson. En effet, si la méthode est d'ordre p ou plus exactement si l'erreur locale est d'ordre $p+1$, on a en supposant $y_n = y(t_n)$ (cf. remarque plus loin) :

$$\overline{y}(t_n+h) = y(t_n+h) - C_n h^{p+1} + O\left(h^{p+2}\right) \tag{5.14}$$

où en général $C_n \neq 0$ et aussi

$$\widehat{y}(t_n+h) = y(t_n+h) - C_n\left(\frac{h}{2}\right)^{p+1} + O\left(h^{p+2}\right). \tag{5.15}$$

On le vérifie aisément pour la méthode d'Euler explicite : dans ce cas $p=1$ et $C_n = \frac{1}{2}y''(t_n)$. Négligeant les termes d'ordre $p+2$ dans (5.14), on a

$$\varepsilon_n(h) \simeq |C_n| h^{p+1}. \tag{5.16}$$

Mais par différence entre (5.15) et (5.14), on obtient aussi :

$$\widehat{y}(t_n+h) - \overline{y}(t_n+h) \simeq C_n\left(\frac{h}{2}\right)^{p+1}\left(2^{p+1}-1\right)$$

d'où une évaluation de l'erreur $\varepsilon_n(h)$ en fonction des valeurs calculées $\widehat{y}(t_n+h)$ et $\overline{y}(t_n+h)$ soit

$$\left|\varepsilon_n\left(\frac{h}{2}\right)\right| \simeq \frac{|\widehat{y}(t_n+h) - \overline{y}(t_n+h)|}{2^{p+1}-1} \text{ (on note } D_n(h) \text{ cette expression).}$$

D'où une première méthode de contrôle du pas : on se donne une tolérance $\alpha > 0$;

$$\left\{\begin{array}{l} \text{étant donné un pas de départ } h \\ \cdot \text{ on calcule } \overline{y}(t_n+h) \text{ en une itération de pas } h \\ \cdot \text{ on calcule } \widehat{y}(t_n+h) \text{ en deux itérations de pas } h/2 \\ \left|\begin{array}{l} \cdot \text{si } D_n(h) > \alpha h, \text{ on recommence avec } h \text{ remplacé par } h/2 \\ \cdot \text{si } D_n(h) < \alpha h, \text{ on accepte } \widehat{y}(t_n+h) \text{ comme valeur approchée de } y(t_n+h) \\ \quad \text{et on continue : } \left\{\begin{array}{l} -\text{avec le même pas } h \text{ si } \alpha' h < D_n(h) < \alpha h, \text{ où } \alpha' \text{ est une} \\ \text{tolérance inférieure (par exemple } \alpha' = \alpha/2^{p+1}) \\ -\text{avec le pas } 2h \text{ si } D_n(h) < \alpha' h. \end{array}\right. \end{array}\right. \end{array}\right. \tag{5.17}$$

Remarque 5.3 L'introduction de α' permet d'allonger le pas pour ne pas effectuer trop d'itérations inutiles.

Un contrôle plus sophistiqué consiste, au lieu de diviser et multiplier par 2, à prendre comme nouveau pas le pas "optimal" h réalisant l'égalité

$$\alpha\overline{h} = \varepsilon_n\left(\overline{h}\right) \simeq C_n\overline{h}^{p+1} \simeq 2^{p+1}\left(\frac{\overline{h}}{h}\right)^{p+1} D_n(h)$$

soit

$$\overline{h} = \frac{h}{2}\left(\frac{\alpha h}{2D_n(h)}\right)^{\frac{1}{p}}.$$

On contrôle alors comme suit : partant du pas h de l'étape précédente
- si $h \simeq 2\overline{h}$, on continue avec $\widehat{y}(t_n + h)$ et le même pas h
- si $h >> 2\overline{h}$, on recommence avec h remplacé par $2\overline{h}$
- si $h << 2\overline{h}$, on continue avec $\widehat{y}(t_n + h)$ et le pas $h^* = \min\left(2\overline{h}, h_0\right)$ où h_0 est une limite supérieure de sécurité pour le pas.

Remarque 5.4 – Dans le raisonnement ci-dessus, nous avons supposé $y_n = y(t_n)$ ce qui bien sûr n'est pas réalisé en pratique. On peut cependant justifier les conclusions ci-dessus. On pourra par exemple les vérifier directement sur la méthode d'Euler.
- S'il s'avère nécessaire de prendre des h trop petits ou si on se heurte à plusieurs échecs successifs, la situation est anormale et il faut procéder à des modifications (ordre insuffisant par rapport à la précision demandée, etc...). On veillera à ce que l'ordre de grandeur de la précision requise soit compatible avec l'ordre de la méthode et la précision de la machine.

5.3 Méthodes à pas multiples

Les méthodes à un pas utilisent seulement la valeur approchée y_n de $y(t_n)$ pour calculer une valeur approchée y_{n+1} de $y(t_{n+1})$. Les méthodes à pas multiples utilisent aussi l'information obtenue aux temps précédents $t_{n-1}, t_{n-2}, ..., t_{n-r}$.

Nous décrirons ici les méthodes d'Adams qui consistent à remplacer $f(t, y(t))$ par un polynôme d'interpolation aux points $t_{n-r}, t_{n-r+1}, ..., t_{n-1}, t_n, (t_{n+1})$, dans le calcul de

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\, dt.$$

Si P_n est le polynôme en question, les valeurs approchées y_n seront obtenues par l'équation approchée

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} P_n(t)\, dt. \tag{5.18}$$

Les formules seront implicites ou explicites selon que t_{n+1} est l'un des points d'interpolation ou non.

5.3.1 Méthodes d'Adams-Bashforth à $r+1$ pas

On suppose connues des valeurs approchées y_n de $y(t_n)$ et $f_n, f_{n-1}, ..., f_{n-r}$ de $f(t, y(t))$ respectivement aux points $t_n, t_{n-1}, ..., t_{n-r}$. Le polynôme P_n est choisi comme le polynôme de degré inférieur ou égal à r tel que

$$P_n(t_{n-i}) = f_{n-i} \qquad \forall i = 0, ..., r.$$

Pour $t \in [t_n, t_{n+1}]$, $t_{n+1} = t_n + h$. Posons $t = t_n + sh$ où $s \in [0, 1]$. Alors comme il a été établi dans le chapitre d'interpolation (cf. 2.9),

$$P_n(t_n + sh) = \sum_{i=0}^{r} \begin{pmatrix} s+i-1 \\ i \end{pmatrix} \Delta^{-i} f_n. \tag{5.19}$$

Ceci est la formule de Newton rétrograde obtenue à l'aide des différences finies rétrograde

$$\Delta^{-i} f_k = \begin{cases} f_k & \text{si } i = 0 \\ \Delta^{-i-1} f_k - \Delta^{-i-1} f_{k-1} & \text{si } i \geq 1 \end{cases}$$

et $\begin{pmatrix} s \\ k \end{pmatrix}$ est le coefficient du binôme généralisé aux valeurs non entières soit

$$\begin{pmatrix} s \\ k \end{pmatrix} = \frac{s(s-1)...(s-k+1)}{1.2...k}.$$

On obtient y_{n+1} à l'aide de (5.18) et (5.19), soit, en effectuant le changement de variables $t = t_n + sh$:

$$y_{n+1} = y_n + \sum_{i=0}^{r} \Delta^{-i} f_n . \int_0^1 \begin{pmatrix} s+i-1 \\ i \end{pmatrix} h\, ds. \tag{5.20}$$

Notons

$$\gamma_i = \int_0^1 \begin{pmatrix} s+i-1 \\ i \end{pmatrix} ds = \int_0^1 \frac{s(s+1)...(s+i-1)}{i!} ds.$$

On montre facilement que les γ_i vérifient la relation

$$\gamma_0 = 1,\ 1 = \frac{\gamma_0}{i+1} + \frac{\gamma_1}{i} + ... + \frac{\gamma_{i-1}}{2} + \gamma_i$$

ce qui permet de les calculer par récurrence. Il est important de noter qu'ils ne dépendent pas de r, ce qui est utile lorsqu'on veut faire varier l'ordre r dans un même calcul. On obtient ainsi

$$\gamma_0 = 1,\ \gamma_1 = \frac{1}{2},\ \gamma_2 = \frac{5}{12},\ \gamma_3 = \frac{3}{8},\ \gamma_4 = \frac{251}{720},\ \gamma_5 = \frac{95}{288}.$$

Dans la pratique, on préfère expliciter la relation (5.20) directement en fonction des valeurs f_{n-i} d'où une formule du type

$$y_{n+1} = y_n + h \sum_{i=0}^{r} b_{i,r} f_{n-i}.$$

A l'aide de la définition des différences divisées, on vérifie

$$b_{r,r} = (-1)^r \gamma_r,\ b_{i,r} = b_{i,r-1} + (-1)^i \left(\begin{array}{c} r \\ i \end{array} \right) \gamma_r,\ 0 \leq i \leq r.$$

On obtient ainsi le tableau suivant :

	$b_{0,r}$	$b_{1,r}$	$b_{2,r}$	$b_{3,r}$	$b_{4,r}$	$b_{5,r}$	$b_{6,r}$	γ_r
$r=0$	1							1
$r=1$	$\frac{3}{2}$	$-\frac{1}{2}$						$\frac{1}{2}$
$r=2$	$\frac{23}{12}$	$-\frac{4}{3}$	$\frac{5}{12}$					$\frac{5}{12}$
$r=3$	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{3}{8}$				$\frac{3}{8}$
$r=4$	$\frac{1901}{720}$	$-\frac{1387}{360}$	$\frac{109}{30}$	$-\frac{637}{360}$	$\frac{251}{720}$			$\frac{251}{720}$
$r=5$	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{4991}{720}$	$-\frac{3649}{720}$	$\frac{959}{480}$	$-\frac{95}{288}$		$\frac{95}{288}$
$r=6$	$\frac{199441}{60840}$	$-\frac{18817}{2520}$	$\frac{238783}{20160}$	$-\frac{10979}{945}$	$\frac{139313}{20160}$	$-\frac{5783}{2520}$	$\frac{19807}{60840}$	$\frac{19807}{60840}$

On utilise la méthode d'Adams-Bashforth à 4 pas, le plus souvent, soit :

$$y_{n+1} = y_n + \frac{h}{24} \left(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3} \right). \tag{5.21}$$

On montre que cette méthode est d'ordre 4 (ie. l'erreur globale de consistance est en h^4) et qu'elle est stable (en un sens analogue à (4.17)) et ce sous des hypothèses naturelles de régularité de f. Cependant, les constantes de stabilité sont souvent grandes ce qui fait qu'on constate souvent une instabilité numérique analogue à celle que nous avons souligné comme étant fréquente dans la méthode d'Euler explicite (cf. paragraphe 4.2.1). Comme toujours, pour pallier cet inconvénient, on préfère utiliser des méthodes implicites.

5.3.2 Méthodes d'Adams-Moulton à $r+1$ pas

On interpole la fonction $f(t, y(t))$ aux points $t_{n+1},\ t_n,\ ...,\ t_{n-r}$ par le polynôme Q_n de degré inférieur ou égal à $r+1$ tel que

$$\left\{ \begin{array}{l} Q_n(t_{n-i}) = f_{n-i} \quad i = 0, 1, ..., r \\ Q_n(t_{n+1}) = f_{n+1} \quad \text{(valeur encore inconnue).} \end{array} \right.$$

D'après la formule de Newton rétrograde, on a

$$Q_n(t_{n+1} + sh) = \sum_{i=0}^{r+1} \left(\begin{array}{c} s+i-1 \\ i \end{array} \right) \Delta^{-i} f_{n+1}.$$

On obtient alors y_{n+1} à l'aide de

$$y_{n+1} = y_n + h \sum_{i=0}^{r+1} \gamma_i^* \Delta^{-i} f_{n+1} \tag{5.22}$$

où

$$\gamma_i^* = \int_{-1}^{0} \frac{s\,(s+1)\ldots(s+i-1)}{i!} ds,\ i \geq 1,\ \gamma_0^* = 1.$$

On vérifie

$$\gamma_i^* = \gamma_i - \gamma_{i-1},\ i \geq 1.$$

Comme précédemment, on préfère écrire (5.22) sous la forme

$$y_{n+1} = y_n + h \sum_{i=-1}^{r} b_{i,r}^* f_{n-i}$$

où, comme on le vérifie aisément, les $b_{i,r}^*$ satisfont à

$$b_{r,r}^* = (-1)^{r+1} \gamma_{r+1}^*,\ b_{i,r}^* = b_{i,r-1}^* + (-1)^{i+1} \begin{pmatrix} r+1 \\ i+1 \end{pmatrix} \gamma_{r+1}^*.$$

On obtient le tableau

	$b_{-1,r}^*$	$b_{0,r}^*$	$b_{1,r}^*$	$b_{2,r}^*$	$b_{3,r}^*$	$b_{4,r}^*$	$b_{5,r}^*$	$b_{6,r}^*$	$b_{7,r}^*$
$r=0$	$\frac{1}{2}$	$\frac{1}{2}$							1
$r=1$	$\frac{5}{12}$	$\frac{2}{3}$	$-\frac{1}{12}$						$-\frac{1}{2}$
$r=2$	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$					$-\frac{1}{12}$
$r=3$	$\frac{251}{720}$	$\frac{323}{360}$	$-\frac{11}{30}$	$\frac{53}{360}$	$-\frac{19}{720}$				$-\frac{1}{24}$
$r=4$	$\frac{95}{288}$	$\frac{1427}{1440}$	$-\frac{133}{240}$	$\frac{241}{720}$	$-\frac{173}{1440}$	$\frac{3}{160}$			$-\frac{19}{720}$
$r=5$	$\frac{19087}{60480}$	$\frac{2713}{2520}$	$-\frac{15487}{20160}$	$\frac{586}{945}$	$-\frac{6737}{20160}$	$\frac{263}{2520}$	$-\frac{863}{60480}$		$-\frac{3}{160}$
$r=6$	$\frac{36799}{120960}$	$\frac{139849}{120960}$	$-\frac{121797}{120960}$	$\frac{123133}{120960}$	$-\frac{88545}{120960}$	$\frac{41499}{120960}$	$-\frac{11351}{120960}$	$\frac{275}{24192}$	$-\frac{863}{60480}$

on utilise beaucoup la formule d'Adams-Moulton à 3 pas soit

$$y_{n+1} = y_n + \frac{h}{24}\left(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right). \tag{5.23}$$

On montre, sous des hypothèses de régularité, que cette méthode est d'ordre 4 et stable. Les coefficients de stabilité sont bien meilleurs que pour la formule explicite d'ordre 4 (5.21). Bien sûr, il faut "payer le prix" car (5.23) définit y_{n+1} de façon implicite puisque $f_{n+1} = f\left(t_{n+1}, y_{n+1}\right)$. Il faut donc résoudre un système non linéaire. On peut utiliser les méthodes générales de résolution de tels systèmes ou l'idée générale que nous développons dans le paragraphe suivant.

5.3.3 Méthode de prédicteur-correcteur

Pour résoudre (5.23) on peut utiliser une méthode d'approximations successives (ou de point fixe, voir Section 8.2.2.5) consistant à construire la suite $y^0, y^1, y^2, ..., y^p$ définie par

$$\begin{cases} y^{p+1} = y_n + \frac{h}{24}\left(9f\left(t_{n+1}, y^p\right) + 19f_n - 5f_{n-1} + f_{n-2}\right) \\ y^0 \text{ à choisir.} \end{cases} \tag{5.24}$$

On peut mener la suite jusqu'à convergence (en général y^p converge vers y_{n+1} quand p tend vers l'infini). Le plus souvent, on se contente de quelques itérations, voire 1 ou 2. D'autre part, la valeur initiale y^0 est souvent obtenue à l'aide d'un pas d'une méthode explicite de même ordre : il s'agit alors d'une méthode de prédicteur-correcteur : l'évaluation de y^0 correspond à une prédiction ; on corrige ensuite cette valeur à l'aide d'une ou deux itérations de (5.24). Ainsi, le schéma suivant est souvent utilisé :

$$\left\{\begin{array}{l} \text{Prédicteur : Formule d'Adams-Bashforth d'ordre 4} \\ y^0 = y_n + \frac{h}{24}\left(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right) \\ \text{Correcteur : une ou deux itérations de la méthode d'Adams-Moulton d'ordre 4} \\ y^{p+1} = y_n + \frac{h}{24}\left(9f\left(t_{n+1}, y^p\right) + 19f_n - 5f_{n-1} + f_{n-2}\right),\ p = 0, 1. \end{array}\right. \tag{5.25}$$

On montre que la méthode (5.25) est aussi d'ordre 4 ; sa stabilité est nettement meilleure que celle d'Adams-Bashforth ; d'autre part, la résolution du système non linéaire de la formule d'Adams-Moulton est faite de façon "explicite".

Remarque 5.5 cette technique de prédicteur-correcteur peut-être bien sûr utilisée dans de nombreuses autres situations. Ainsi, sans aller jusqu'à l'ordre 4, on peut l'appliquer aux méthodes d'Euler et Crank-Nicholson :

$$\left\{\begin{array}{l} \text{prédicteur : } y^0 = y_n + hf_n \\ \text{correcteur : } y^{p+1} = y_n + \frac{h}{2}\left(f\left(t_{n+1}, y^p\right) + f_n\right),\ p = 0, 1. \end{array}\right. \tag{5.26}$$

Il existe d'autres prédicteurs-correcteurs multipas : citons les formules de Milne :

$$\left\{\begin{array}{l} \text{prédicteur : } y^0 = y_{n-3} + \frac{4h}{3}\left(2f_n - f_{n-1} + 2f_{n-2}\right) \\ \text{correcteur : } y^{p+1} = y_{n-1} + \frac{h}{3}\left(f\left(t_{n+1}, y^p\right) + 4f_n + f_{n-1}\right). \end{array}\right. \tag{5.27}$$

Cette méthode correspond à des intégrations de t_{n-r} à t_{n+1} au lieu de t_n à t_{n+1} de l'équation différentielle. Ici, l'intégration est faite avec une méthode de Simpson. On a une méthode d'ordre 4, mais généralement moins stable que celle d'Adams.

5.4 Comparaison des méthodes

Les méthodes les plus couramment utilisées sont celles de Runge-Kutta d'ordre 4 et les méthodes de prédicteur-correcteur d'Adams d'ordre 4. Les codes les plus modernes utilisent de plus un contrôle du pas ainsi qu'un contrôle de l'ordre à chaque étape, adaptant ainsi le pas et l'ordre à la précision requise et aux irrégularités éventuelles de la solution. Des algorithmes multiples assez sophistiqués utilisant ces idées ont été développés en particulier par Gear et fournissent actuellement les méthodes les plus performantes. Signalons que, pour des problèmes mal conditionnés (dits raides), on doit utiliser des méthodes moins performantes mais plus stables : les plus efficaces sont celles des différentiations rétrogrades que nous ne développerons pas ici. Elles sont évidemment complètement implicites (les itérations éventuelles sont menées jusqu'à la convergence).

En ce qui concerne un choix entre les méthodes de Runge-Kutta et celles d'Adams, signalons les avantages et les inconvénients de chacune des méthodes :

– Runge-Kutta : c'est une méthode explicite, relativement stable et précise, facile à implémenter et aisément adaptable à un pas variable (donc contrôle de pas aisé). De plus, elle est "self-starting", c'est-à-dire qu'une seule valeur $y^0 \simeq y(0)$ suffit pour l'initier.

Elle présente cependant deux désavantages importants : chaque itération requiert 4 évaluations de fonctions (pour atteindre l'ordre 4). De plus, on n'a aucun contrôle effectif de l'erreur locale ce qui rend difficile un bon contrôle du pas. On peut pallier ce dernier point en couplant deux méthodes, mais ceci augmente encore le coût.

– Adams : pour une précision d'ordre 4, une seule évaluation de fonction est nécessaire à chaque pas dans la méthode explicite, ce qui est bien sûr remarquable. On perd en stabilité par rapport à Runge-Kutta, mais on peut stabiliser en utilisant un correcteur. Ceci augmente le coût mais permet aussi, par comparaison, une estimation de l'erreur locale.

Les difficultés viennent d'une part de l'initiation : pour les méthodes à 4 pas, 4 valeurs sont nécessaires pour initier la méthodes. On peut les obtenir par une méthode de Runge-Kutta appliquée plusieurs fois. on peut aussi prendre une méthode à nombre de pas variables en commençant par un pas et montant progressivement à 4. Il faut alors faire un contrôle sérieux de l'erreur.

Un autre désavantage est que le fait de changer de pas apporte un surcoût important par la nécessité de recalculer les coefficients de la formule. Ceux que nous avons donnés (les $b_{i,r}$) correcpondent à un pas constant. Ils sont extrêmement plus compliqués si les $t_{n+1}, t_n, t_{n-1}, ..., t_{n-r}$ correspondent à des accroissements variables. On préfère alors modifier l'ordre (i.e. le nombre de pas) ce qui donne un surcoût de calcul très réduit. On prend alors le risque d'augmenter l'instabilité...

Conclusion : L'écriture d'un bon programme de résolution de systèmes différentiels passe par la bonne compréhension des algorithmes élémentaires développés dans les paragraphes précédents. On peut à partir de là écrire des algorithmes performants adaptés aux situations particulières des utilisateurs en retenant les idées générales suivantes :

– les méthodes implicites sont plus stables mais plus coûteuses que les explicites.
– augmenter l'ordre augmente aussi l'instabilité.
– un contrôle du pas est généralement nécessaire, mais augmente considérablement le coût du calcul.
– les problèmes raides (mal conditionnés) doivent être abordés avec beaucoup de circonspection et des algorithmes spécifiques.

5.5 Applications à des problèmes aux limites

Soit à résoudre par exemple :

$$\begin{cases} y'' = y^3 + y + 1 \text{ sur } (0,1) \\ y(0) = a,\ y(1) = b,\ a \text{ et } b \text{ donnés.} \end{cases}$$

Il ne s'agit pas ici d'une équation différentielle ordinaire puisque la solution ne dépend plus seulement de "l'état initial" en $t = 0$, mais aussi de $y(1)$ qui est donné. Il s'agit d'un "problème aux limites", donc de nature tout à fait différente. On peut cependant le résoudre à l'aide des techniques précédentes couplées avec une méthode de tir.

Commençons par résoudre le système différentiel ordinaire

$$\begin{cases} y''_\alpha = y_\alpha^3 + y_\alpha + 1,\ t > 0 \\ y_\alpha(0) = a \\ y'_\alpha(0) = \alpha \text{ paramètre.} \end{cases}$$

Ce problème entre dans le cadre précédent en posant $U(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}$ comme il a été vu en début de chapitre. On peut donc le résoudre à l'aide des méthodes décrites plus haut. On essaie alors de déterminer le paramètre α pour que $y_\alpha(1) = b$. La Figure 5.1 illustre

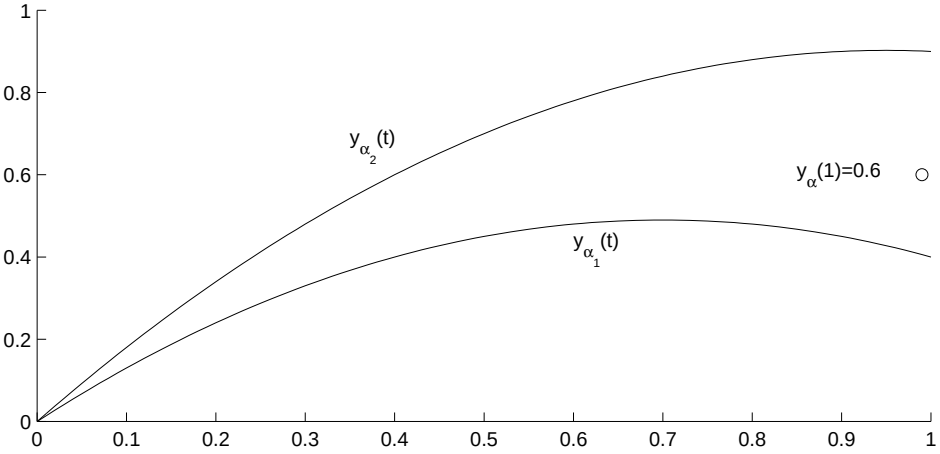

FIGURE 5.1 – La méthode du tir pour résoudre un problème aux limites

bien la terminologie "méthode de tir" : on essaie de "viser" assez juste pour que $y_\alpha(1) = b$.

La valeur $y_\alpha(1)$ est une fonction non linéaire de α et le problème consiste à résoudre l'équation non linéaire $y_\alpha(1) - b = 0$. Partant de deux valeurs initiales $\alpha_0,\ \alpha_1$, on peut alors lui appliquer la méthode de la sécante (voir chapitre 8), soit

$$\alpha_{n+1} = \alpha_n - (y_{\alpha_n}(1) - b)\frac{\alpha_n - \alpha_{n-1}}{y_{\alpha_n}(1) - y_{\alpha_{n-1}}(1)}.$$

Evidemment, chaque calcul de $y_\alpha(1)$ nécessite la résolution de l'équation différentielle le long de $(0,1)$. Cette méthode s'avère donc très vite coûteuse. Elle peut cependant être convenable dans certaines situations et l'idée est à retenir.

5.6 Exercices du chapitre 5

Exercice 5.1 Soit a, c, b_1, b_2 des réels avec $0 \leq b_1 \leq b_2 \leq 1$. On introduit un schéma de type Runge-Kutta par les formules

$$\begin{cases} y_{n+1} = y_n + h_n(b_1 k_1^n + b_2 k_2^n) \\ k_1^n = f(t_n, y_n) \\ k_2^n = f(t_n + ch_n, y_n + ah_n k_1^n) \end{cases}$$

1) Que doit on choisir comme valeurs pour les coefficients a, c, b_1, b_2 si on souhaite que ce schéma numérique soit d'ordre deux ?
2) Dans ce cas, retrouve-t-on nécessairement la méthode de Runge-Kutta d'ordre deux (5.9) ?

Exercice 5.2 Pour résoudre l'équation différentielle

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(0) = y_0 \end{cases}$$

(avec f lipschitzienne par rapport à la deuxième variable), on envisage une méthode à pas multiple :

$$y_{n+1} = y_n + h_n \left(\beta_0 f_{n-2} + \beta_1 f_{n-1} + \beta_2 f_n\right).$$

Déterminer les coefficients $\beta_0, \beta_1, \beta_2$ pour que la méthode soit d'ordre maximum. Est-elle alors stable, consistante, convergente ?

Exercice 5.3 Soit β un nombre réel strictement positif donné. On considère l'équation différentielle

$$\begin{cases} y'(t) = -\beta y(t) \quad t > 0 \\ y(0) = y_0 \end{cases} \tag{5.28}$$

où y_0 est également donné. Soit $h > 0$ le pas (fixe) de la subdivision, pour les méthodes de Runge-Kutta deux (5.9) et Runge-Kutta quatre (5.11), on note y_n la valeur approchée de $y(nh)$.
1) Quelle est la solution exacte de (5.28) ? Est-il vrai que $\lim_{t\to+\infty} y(t) = 0$?
2) Pour chacune des méthodes, à quelle condition sur h a-t-on

$$\lim_{n\to+\infty} y_n = 0?$$

Chapitre 6

Résolution de systèmes linéaires - méthodes directes

Les deux problèmes fondamentaux de l'analyse numérique matricielle sont
- la résolution de systèmes linéaires, i.e. la recherche de vecteurs X solutions de

$$AX = b$$

  où A est une matrice -le plus souvent carrée- à coefficients réels (ou complexes) et b un vecteur donné
- le calcul de valeurs propres et de vecteurs propres d'une matrice, i.e. la recherche des scalaires λ (réels ou complexes) et des vecteurs non nuls X tels que

$$AX = \lambda X,$$

  A matrice carrée donnée.

Nous traitons dans ce chapitre le premier type de problèmes. Nous n'abordons pas la seconde question mais des livres de référence le font.

6.1 Quelques remarques générales

6.1.1 Origines des problèmes

Elles sont très variées et apparaissent déjà en plusieurs endroits dans les autres chapitres du cours d'analyse numérique. Citons quelques-uns des problèmes conduisant à la résolution de systèmes linéaires :
- Résolution d'équations aux dérivées partielles (ceci en constitue une source très importante. Voir quelques exemples au paragraphe 7.2.4)
- Approximation au sens des moindres carrés
- Calcul de fonctions splines
- Programmation linéaire (méthode du simplexe, d'Uzawa)
- Algorithmes d'optimisation non linéaire
- etc...

6.1.2 Méthodes utilisées

Elles sont de deux types
- les méthodes directes : celles où on obtient la valeur exacte de la solution (aux erreurs d'arrondi près) en un nombre fini d'opérations,
- les méthodes itératives : elles consistent à construire une suite de vecteurs X^k convergeant vers la solution X cherchée. On s'arrête bien sûr au bout d'un nombre fini n d'itérations choisi pour que X^n soit suffisamment voisin de X.

Remarque 6.1 On n'utilise jamais les formules de Cramer car elles nécessitent le calcul de déterminants qui requièrent un nombre trop important d'opérations élémentaires.

Remarque 6.2 On ne calcule pas A^{-1} pour résoudre $AX = b$. Numériquement, c'est le contraire qui se produit : le calcul de A^{-1}, ainsi d'ailleurs que celui de $\det A$, sont des sous-produits des méthodes de résolution de systèmes. Ainsi, le calcul de A^{-1} s'effectuera en résolvant successivement les systèmes

$$\{AX_i = e_i,\ i = 1, ..., N\},$$

où e_i est le ième vecteur de la base canonique de $\mathbb{R}^N$. La ième colonne de A^{-1} est donnée par X_i comme on le vérifie aisément.

6.1.3 Structure de la matrice

La structure de la matrice a une incidence fondamentale sur la difficulté de la résolution et sur le choix de la méthode

6.1.3.1 Cas d'une matrice diagonale

Tous les éléments sont nuls sauf ceux de la diagonale principale :

$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_N \end{bmatrix}$$

La résolution de $AX = b$ est alors immédiate. Si $X = (x_1, ..., x_{\mathrm{N}})$ et $b = (b_1, ..., b_{\mathrm{N}})$, on a

$$x_i = \frac{b_i}{a_i},\ i = 1, ..., N$$

ceci en supposant bien sûr que tous les a_i sont non nuls.

6.1.3.2 Matrice triangulaire supérieure (ou inférieure)

Tous les éléments au-dessous (ou au-dessus) de la diagonale sont nuls

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1\mathrm{N}} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{\mathrm{N}\,\mathrm{N}} \end{bmatrix}$$

Dans ce cas on utilisera toujours une méthode directe de résolution par remontée soit :

$$\left\{\begin{array}{r} a_{11}x_1 + a_{12}x_2 + ... + a_{1\mathrm{N}}x_{\mathrm{N}} = b_1 \\ a_{22}x_2 + ... + a_{2\mathrm{N}}x_{\mathrm{N}} = b_2 \\ ..... \\ a_{\mathrm{N}-1,\mathrm{N}-1}x_{\mathrm{N}-1} + a_{\mathrm{N}-1,\mathrm{N}}x_{\mathrm{N}} = b_{\mathrm{N}-1} \\ a_{\mathrm{N}\mathrm{N}}x_{\mathrm{N}} = b_{\mathrm{N}} \end{array}\right. \tag{6.1}$$

On commence par résoudre la dernière équation ; on substitue le résultat obtenu pour x_{N} dans la précédente, ce qui permet de calculer $x_{\mathrm{N}-1}$, etc...

$$\left\{\begin{array}{l} x_{\mathrm{N}} = b_{\mathrm{N}} / a_{\mathrm{N}\mathrm{N}} \\ x_{\mathrm{N}-1} = \left(b_{\mathrm{N}-1} - a_{\mathrm{N}-1,\mathrm{N}}x_{\mathrm{N}}\right) / a_{\mathrm{N}-1,\mathrm{N}-1} \\ ..... \\ x_i = \left(b_i - a_{i,\mathrm{N}}x_{\mathrm{N}} - a_{i,\mathrm{N}-1}x_{\mathrm{N}-1} - ... - a_{i,i+1}x_{i+1}\right) / a_{ii} \end{array}\right. \tag{6.2}$$

Il est intéressant de compter déjà ici le nombre d'opérations élémentaires (i.e. addition, multiplication, division) nécessaires pour mener à bien le calcul de tous les x_i. Chaque x_i nécessite (une fois x_{i+1}, x_{i+2}, ..., x_{N} calculés) 1 division, $N - i$ multiplications, $N - i$ additions.
Puisque

$$\sum_{i=N}^{1} (N - i) = \sum_{j=0}^{N-1} j = \frac{N\,(N-1)}{2},$$

il faut au total N divisions, $\frac{N(N-1)}{2}$ multiplications, $\frac{N(N-1)}{2}$ additions, soit N^2 opérations élémentaires.
Le cas des matrices triangulaires inférieures se traite de façon identique par un procédé de descente.

6.1.3.3 Autres cas

Dans le cas général, les méthodes de résolution seront d'autant plus simples que les deux phénomènes suivants seront plus importants :
- la matrice est creuse (c'est-à-dire a beaucoup de zéros)
- les termes non nuls de la matrice sont localisés et en particulier sont rassemblés autour de la diagonale. Ceci conduit à la notion de matrice-bande.

Exemples :
matrice tridiagonale

$$A = \begin{bmatrix} \bullet & \bullet & 0 & \cdots & 0 \\ \bullet & \bullet & \bullet & \ddots & \vdots \\ 0 & \bullet & \bullet & \bullet & 0 \\ \vdots & \ddots & \bullet & \bullet & \bullet \\ 0 & \cdots & 0 & \bullet & \bullet \end{bmatrix}$$

Tous les éléments sont nuls sauf les $a_{i,j}$ pour lesquels $i - 1 \leq j \leq i + 1$. Nous verrons plus loin une méthode directe de résolution du système avec un nombre d'opérations $8N - 6$, donc très faible.

matrice pentadiagonale :

$$A = \begin{bmatrix} \bullet & \bullet & \bullet & 0 & \cdots & 0 \\ \bullet & \bullet & \bullet & \bullet & \ddots & \vdots \\ \bullet & \bullet & \bullet & \bullet & \bullet & 0 \\ 0 & \bullet & \bullet & \bullet & \bullet & \bullet \\ \vdots & \ddots & \bullet & \bullet & \bullet & \bullet \\ 0 & \cdots & 0 & \bullet & \bullet & \bullet \end{bmatrix}$$

matrice diagonale par blocs

$$A = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & A_{33} & \\ & & & A_{44} \end{bmatrix}$$

Dans ce cas le système associé se réduit à plusieurs "petits" systèmes découplés de dimension réduite (mais pas nécessairement égale), soit

$$\{A_{ii}X_i = b_i,\ i = 1, 2, 3, 4\}$$

où $X = (X_1, X_2, X_3, X_4)\,, b = (b_1, b_2, b_3, b_4)\,.$

On définit de façon analogue les matrices tridiagonales par blocs, auxquelles on étend les techniques développées pour les matrices tridiagonales au sens ci-dessus : celles-ci sont alors appelées tridiagonales par points et apparaissent comme des cas particuliers des tridiagonales par blocs.

6.1.3.4 Matrices symétriques

Une autre propriété de la matrice peut jouer un rôle important, c'est la symétrie. On dit qu'une matrice A est symétrique [1] si elle coincide avec sa transposée, c'est-à-dire si

$$a_{ij} = a_{ji} \qquad \forall i, j.$$

Un sous-ensemble de ces matrices donnera lieu à une étude particulière, c'est celui des matrices symétriques définies positives. On dit qu'une matrice symétrique est positive si

$$\forall X \in \mathbb{R}^N \qquad {}^tXAX \geq 0$$

($X \longrightarrow {}^tXAX$ est une forme quadratique). Elle est de plus définie positive si ${}^tXAX$ n'est nul que si X est nul.

Exemple : $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ est symétrique définie positive car :

$$\begin{aligned} \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 2x & -y \\ -x & 2y \end{bmatrix} \\ &= 2\left(x^2 - xy + y^2\right) \\ &= 2\left[\left(x - \frac{1}{2}y\right)^2 + \frac{3}{4}y^2\right] > 0 \text{ si } (x, y) \neq (0, 0)\,. \end{aligned}$$

1. voir section 6.5

La matrice $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ est symétrique positive, non définie car

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} x & -y \\ -x & y \end{bmatrix}$$
$$= x^2 - 2xy + y^2 = (x-y)^2 \geq 0$$

mais $(x-y)^2$ s'annule si $x = y$.

Des méthodes particulières seront développées pour ces matrices qui apparaissent très souvent dans la discrétisation de problèmes variationnels (minimisation de fonctionnelles).

6.1.4 Stockage des matrices

Le stockage de tous les éléments d'une matrice d'ordre N requiert bien sûr en mémoire le stockage de N^2 nombres réels. La discrétisation de certains problèmes, en particulier les équations aux dérivées partielles en dimension 2 ou 3 conduisent rapidement à des matrices d'ordre N égal à plusieurs millions ; d'où la nécessité "d'optimiser" le stockage des matrices. Indiquons ici quelques techniques souvent utilisées.

- Si la matrice est symétrique, on ne stockera bien sûr que les éléments diagonaux et surdiagonaux soit au total $\frac{N(N+1)}{2}$ si la dimension est N.
- De très grands systèmes (plusieurs millions d'équations) peuvent être facilement traités si les matrices sont creuses (sparse en anglais) comme c'est souvent le cas pour celles provenant de la discrétisation des équations aux dérivées partielles. On obtient alors le plus souvent des matrices-bandes dont la largeur de bande est petite devant la dimension :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1b} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2b} & a_{2,b+1} & \ddots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & 0 \\ a_{b1} & a_{b2} & \cdots & a_{bb} & a_{b,b+1} & \ddots & \ddots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & a_{\mathrm{N},\mathrm{N}-b+1} & \cdots & \cdots & a_{\mathrm{NN}} \end{bmatrix}$$

avec $b \ll N$ (par exemple $b = 100$ et $N = 10000$). On essaiera dans la mesure du possible de ne stocker que les éléments non nuls. On peut par exemple utiliser un tableau "redressé" comme suit :

$$M = \begin{bmatrix} 0 & & & a_{11} & a_{12} & \cdots & a_{1b} \\ & & a_{21} & a_{22} & a_{23} & \cdots & a_{2,b+1} \\ a_{b1} & a_{b2} & \cdots & \cdots & \cdots & \cdots & a_{b,2b-1} \\ \vdots & & & & & & 0 \\ \vdots & & & & & & \vdots \\ \vdots & & & & & & \vdots \\ a_{\mathrm{N},\mathrm{N}-b+1} & \cdots & a_{\mathrm{NN}} & 0 & \cdots & \cdots & 0 \end{bmatrix}$$

tableau de dimension $(2b-1) \times N << N^2$ si $b << N$, $a_{IJ} = m_{ij}$ si $\begin{cases} i = I \\ j = J - I + b \end{cases}$.

Cette méthode conduit cependant à mémoriser $b(b-1)$ valeurs nulles inutiles. On lui préfère plutôt la méthode suivante :

– matrice profil (ou méthode de la "ligne de ciel") : elle consiste à tracer une ligne fermée contenant les éléments non nuls de la matrice et laissant à l'extérieur le plus de zéros possibles.

Commençons par le cas d'une matrice symétrique.

$$A = \begin{bmatrix} 7 & . & . & . & . & . \\ 2 & 11 & . & . & . & . \\ 0 & 1 & 3 & . & . & . \\ 0 & 0 & 4 & -1 & . & . \\ 6 & 3 & 0 & 2 & -3 & . \\ 0 & 0 & 0 & 12 & 0 & -7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & & & & \\ 2 & 3 & & & & \\ & 4 & 5 & & & \\ & & 6 & 7 & & \\ 8 & 9 & 10 & 11 & 12 & \\ & & & 13 & 14 & 15 \end{bmatrix} \quad \text{tableau des positions}$$

Le stockage de A est réalisé à l'aide de deux tableaux M et P où M contient les éléments de la matrice à l'intérieur de la ligne de profil et P est un tableau de pointeurs de dimension $N+1$ avec $P(1) = 0$, $P(i+1) =$ adresse dans M du ième élément diagonal de A. Ces deux tableaux caractérisent A compte-tenu que les éléments de A sont rangés ligne par ligne et pour une ligne donnée dans l'ordre croissant des colonnes, ce qui donne dans l'exemple ci-dessus :

$$\begin{array}{lcccccccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ M = & (7 & 2 & 11 & 1 & 3 & 4 & -1 & 6 & 3 & 0 & 2 & -3 & 12 & 0 & -7) \\ P = & (0 & 1 & 3 & 5 & 7 & 12 & 15) & & & & & & & & \end{array}$$

la donnée de M et P permet de retrouver A à l'aide de la formule suivante qu'on obtient de façon élémentaire :

$$\begin{cases} \text{Pour } i = 1, ..., N, \text{ pour } j = i + 1 - (P(i+1) - P(i)), ...i \\ \qquad a_{ij} = M(P(i+1) + j - i). \end{cases}$$

On utilise le même principe pour une matrice non symétrique, mais le profil sera toujours choisi symétrique.

$$A = \begin{bmatrix} 7 & 3 & 0 & 0 & 0 & 0 \\ 2 & 11 & -1 & 0 & 4 & 0 \\ 0 & 1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & -1 & -2 & 3 \\ 6 & 3 & 0 & 2 & -3 & 1 \\ 0 & 0 & 0 & 12 & 0 & -7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & . & . & 15 & . \\ 2 & 4 & 6 & . & 16 & . \\ . & 5 & 7 & 9 & 17 & . \\ . & . & 8 & 10 & 18 & 22 \\ 11 & 12 & 13 & 14 & 19 & 23 \\ . & . & . & 20 & 21 & 24 \end{bmatrix}$$ tableau des positions

La marice A est stockée dans le tableau M suivant la règle du tableau des positions ci-dessus : c'est-à-dire de $i = 1$ à N on parcourt la ligne i jusqu'à l'élément diagonal (exclu), puis on descend la colonne i jusqu'à l'élément diagonal (inclus). Ici encore $P(i+1)$ est l'adresse du ième élément diagonal. Avec l'exemple ci-dessus, on a donc :

$$\begin{aligned} M &= (7\ 2\ 3\ 11\ 1\ -1\ 3\ 4\ 0\ -1\ 6\ 3\ 0\ 2\ 0\ 4\ 0\ -2\ -3\ 12\ 0\ 3\ 1\ -7) \\ P &= (0\ 1\ 4\ 7\ 10\ 19\ 24) \end{aligned}$$

On retrouve A à l'aide de M et P par la règle

$$i - \frac{P(i+1) - P(i) + 1}{2} < j < i,\ a_{ij} = M\left(\frac{P(i+1) + P(i) + 1}{2} + j - i\right)$$

$$i \leq j < i + \frac{P(i+1) - P(i) + 1}{2},\ a_{ij} = M(P(i+1) + i - j).$$

Remarque 6.3 Le gain en mémoire occupé n'est évidemment pas probant dans les exemples numériques ci-dessus. Cependant, un calcul simple à l'aide des formules ci-dessus montre que le gain est important pour des matrices-bandes de grandes dimensions.

Si on veut se débarrasser de tous les éléments non nuls, on peut stocker la matrice sous forme de
matrice morse : on remplace la matrice A par trois tableaux M, P_1, P_2 où on a (par exemple) la règle suivante :
- M est le tableau des éléments non nuls de A
- P_1 est un tableau de pointeurs de dimension $N+1$ avec $P_1(1) = 0$ et $P_1(i+1) =$ adresse du ième coefficient diagonal dans M et
- P_2 est un tableau de pointeurs de dimension égale au nombre d'éléments non nuls de A avec $P_2(k) =$ numéro de la colonne de $M(k)$.

Si M est symétrique, la partie triangulaire inférieure (par exemple) est stockée ligne par ligne de la gauche vers la droite. Si M est quelconque, les coefficients sont rangés ligne par ligne de gauche à droite, le coefficient diagonal étant le dernier.
Dans les deux exemples numériques ci-dessus, on obtient les tableaux de positions suivants

$$\begin{bmatrix} 1 & . & . & . & . & . \\ 2 & 3 & . & . & . & . \\ . & 4 & 5 & . & . & . \\ . & . & 6 & 7 & . & . \\ 8 & 9 & . & 10 & 11 & . \\ . & . & . & 12 & . & 13 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 & . & . & . & . \\ 3 & 6 & 4 & . & 5 & . \\ . & 7 & 8 & . & . & . \\ . & . & 9 & 12 & 10 & 11 \\ 13 & 14 & . & 15 & 17 & 16 \\ . & . & . & 18 & . & 19 \end{bmatrix}$$

On en déduit les tableaux M, P_1, P_2. On montre facilement que la connaissance de ces trois tableaux donne A.

6.2 Méthodes directes de résolution de $AX = b$

6.2.1 Méthode d'élimination de Gauss

Principe : il s'agit de la méthode élémentaire d'élimination des variables. Rappelons-la d'abord sur un exemple simple :

$$(S)\begin{cases} ax+by+cz=d \\ a'x+b'y+c'z=d' \\ a''x+b''y+c''z=d'' \end{cases} \quad \text{soit} \quad \begin{cases} P_1(x,y,z)=d \\ P_2(x,y,z)=d' \\ P_3(x,y,z)=d'' \end{cases}$$

Supposons $a \neq 0$: on peut alors éliminer x dans les deuxième et troisième équations en remplaçant (S) par le système équivalent

$$(S_1)\begin{cases} P_1 = d \\ P_2 - \dfrac{a'}{a}P_1 = d' - \dfrac{a'}{a}d \\ P_3 - \dfrac{a''}{a}P_1 = d'' - \dfrac{a''}{a}d \end{cases},$$

qui est de la forme

$$(S_1)\begin{cases} P_1 = d \\ Q_2 = d_2 \\ Q_3 = d_3 \end{cases} \text{ où } \quad \begin{cases} Q_2 = b_2y + c_2z \\ Q_3 = b_3y + c_3z \end{cases}.$$

Supposons $b_2 \neq 0$: on peut alors éliminer y dans la dernière équation en remplaçant (S_1) par le système équivalent

$$(S_2)\begin{cases} P_1 = d \\ Q_2 = d_2 \\ Q_3 - \dfrac{b_3}{b_2}Q_2 = d_3 - \dfrac{b_3}{b_2}d_2 \end{cases} \quad \text{soit} \quad \begin{cases} ax+by+cz=d \\ b_2y+c_2z=d_2 \\ c_3'z=d_3' \end{cases}$$

qui est un système triangulaire ; on peut le résoudre à l'aide de la méthode de remontée signalée au paragraphe précédent.

La méthode générale est absolument la même. La seule difficulté technique qui se présente est que l'un des pivots utilisés (a, b_2, c_3' dans le calcul précédent) peut être nul auquel cas la méthode échoue. Même s'il n'est pas nul mais très petit, ceci peut conduire à des erreurs très importantes. On peut s'en convaincre à l'aide de l'exemple simple suivant : soit à résoudre :

$$(S)\begin{cases} 10^{-4}x + y = 1 \\ x + y = 2 \end{cases}.$$

La solution "exacte" de (S) est :

$$x = 1,00010... \simeq 1, \quad y = 0,999990... \simeq 1.$$

Prenant 10^{-4} comme pivot, on est ramené au système équivalent :

$$\begin{cases} 10^{-4}x + y = 1 \\ (x+y) - \frac{1}{10^{-4}}\left(10^{-4}x + y\right) = 2 - \frac{1}{10^{-4}} \end{cases} \iff \begin{cases} 10^{-4}x + y = 1 \\ \left(10^4 - 1\right)y = 10^4 - 2 \end{cases}.$$

Supposons qu'on travaille avec trois chiffres significatifs : le système est alors équivalent à

$$\left\{ \begin{array}{l} 10^{-4}x + y = 1 \\ 9990y = 9990 \end{array} \right. \text{ soit } \left\{ \begin{array}{l} x = 0 \\ y = 1 \end{array} \right. !$$

Au contraire, un échange préalable des deux équations donnera un résultat satisfaisant même avec seulement trois chiffres significatifs :

$$\begin{array}{l} \left\{ \begin{array}{c} x + y = 2 \\ 10^{-4}x + y = 1 \end{array} \right. \iff \left\{ \begin{array}{c} x + y = 2 \\ 10^{-4}x + y - 10^{-4}(x + y) = 1 - 2.10^{-4} \end{array} \right. \\ \iff \left\{ \begin{array}{c} x + y = 2 \\ 0,999y = 0,999 \end{array} \right. \iff \left\{ \begin{array}{l} x = 1 \\ y = 1 \end{array} \right. ! \end{array}$$

Nous allons décrire le passage de l'étape p à l'étape $p+1$ dans la méthode de Gauss pour la résolution de

$$AX = b.$$

L'élimination sucessive des variables conduit à un système équivalent

$$A_p X = b^p$$

où A_p est de la forme

$$A_p = \left[\begin{array}{ccccccccc} a_{11}^p & a_{12}^p & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1\mathrm{N}}^p \\ 0 & a_{22}^p & \cdots & \cdots & \cdots & \cdots & \cdots & a_{2\mathrm{N}}^p \\ \vdots & \ddots & \ddots & & & & & \vdots \\ \vdots & & \ddots & \ddots & & & & \vdots \\ \vdots & & & \ddots & \ddots & a_{pp}^p & \cdots & a_{p,\mathrm{N}}^p \\ \vdots & & & & 0 & a_{p+1,p}^p & \cdots & a_{p+1,\mathrm{N}}^p \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & a_{\mathrm{N},p}^p & \cdots & a_{\mathrm{NN}}^p \end{array} \right].$$

Le passage à l'étape suivante consiste à utiliser la p-ième ligne (c'est-à-dire la p-ième équation) pour faire apparaître des zéros à la place de $a_{i,p}^p$, $i = p+1, ...N$, c'est-à-dire éliminer la variable x_p dans les $N - p$ dernières équations). Auparavant, il faut s'assurer que a_{pp}^p n'est pas nul ni même trop petit. Pour cela, on adopte généralement une stratégie du pivot.

– Stratégie du pivot partiel : on détermine l'élément a_{ip}^p (ou l'un des éléments a_{ip}^p) tel que $p \leq i \leq N$ et

$$\left| a_{ip}^p \right| = \max_{p \leq k \leq N} \left| a_{kp}^p \right|.$$

On permute alors les lignes d'indices i et p pour amener en position de pivot l'élément a_{ip}^p.

– Stratégie du pivot total : on détermine l'élément a_{ij}^p (ou l'un des éléments a_{ij}^p) tel que $p \leq i, j \leq N$ et

$$\left| a_{ij}^p \right| = \max_{p \leq l,k \leq N} \left| a_{l,k}^p \right|.$$

On effectue alors des permutations de lignes et de colonnes (c'est-à-dire une permutation sur les inconnues) pour amener en position de pivot l'élément $a_{i,j}^p$.

Remarque 6.4 on se contente le plus souvent de la stratégie du pivot partiel l'autre étant réservée aux systèmes particulièrement délicats ou susceptibles de l'être.

Notons que, si la matrice est inversible, la stratégie de pivot partiel donne toujours un pivot non nul. En effet, on vérifie immédiatement que

$$\det A_p = a_{11}^p . a_{22}^p ... a_{p-1,p-1}^p . \det A'_p$$

où

$$A'_p = \begin{bmatrix} a_{pp}^p & \cdots & a_{p\mathrm{N}}^p \\ \vdots & & \vdots \\ a_{\mathrm{N}p}^p & \cdots & a_{\mathrm{NN}}^p \end{bmatrix}.$$

$\det A_p \neq 0 \Longrightarrow \det A'_p \neq 0 \Longrightarrow$ la colonne $\begin{pmatrix} a_{pp}^p \\ \vdots \\ a_{\mathrm{N}p}^p \end{pmatrix}$ n'est pas égale au vecteur nul.

Supposons maintenant qu'une stratégie du pivot ait été effectuée et que (en gardant les mêmes notations pour A_p)

$$a_{pp}^p \neq 0.$$

Pour obtenir A_{p+1}, on conserve la p-ième équation

$$a_{pp}^p x_p + a_{p,p+1}^p x_{p+1} + ... + a_{p,N}^p x_{\mathrm{N}} = b_p^p$$

et on remplace chaque équation

$$a_{ip}^p x_p + a_{i,p+1}^p x_{p+1} + ... + a_{i,\mathrm{N}}^p x_{\mathrm{N}} = b_i^p,\ i = p+1, ...N$$

par l'équation obtenue en lui retranchant la p-ième multipliée par a_{ip}^p / a_{pp}^p. On obtient ainsi la nouvelle matrice A_{p+1} définie par

$$\begin{aligned} a_{i,j}^{p+1} &= a_{i,j}^p - \frac{a_{ip}^p}{a_{pp}^p} a_{p,j}^p && (6.3) \\ p+1 &\leq i \leq N && (6.4) \\ p+1 &\leq j \leq N. && (6.5) \end{aligned}$$

Les lignes $i = 1$ à p ne sont pas modifiées et les autres coefficients sont nuls. Le second membre est transformé de façon identique

$$\begin{aligned} b_i^{p+1} &= b_i^p - \frac{a_{ip}^p}{a_{pp}^p} b_p^p && (6.6) \\ p+1 &\leq i \leq N. && (6.7) \end{aligned}$$

Dans la pratique, on assimile le vecteur b^p à une $(N+1)$-ème colonne de A_p en posant $a_{i,N+1}^p := b_i^p$. On obtient ainsi un traitement global. On peut d'ailleurs ajouter plusieurs colonnes ce qui permet de résoudre plusieurs systèmes simultanément.
Une fois la triangulation terminée, on résout le système par la méthode de remontée.

Remarque 6.5 comme sous-produit, on obtient un calcul simple du déterminant de A, à savoir

$$\det A = \pm a_{11}^1 a_{22}^2 ... a_{\mathrm{NN}}^N$$

où $\pm$ dépend du nombre de permutations éventuelles de lignes. On vérifie en effet que le déterminant de A_p n'est pas modifié dans le procédé d'élimination (ou de triangulation).

6.2.1.1 Calcul du nombre d'opérations élémentaires

Pour passer de A_p à A_{p+1} on a
$N-p$ divisions
$(N-p)(N-p+1)$ additions
$(N-p)(N-p+1)$ multiplications, soit

$$\begin{aligned}\sum_{p=1}^{N-1}(N-p) &= \frac{N(N-1)}{2} \text{ divisions}\\ \sum_{p=1}^{N-1}(N-p)^2+(N-p) &= \frac{N(N-1)(2N-1)}{6}+\frac{N(N-1)}{2}\\ &= \frac{N^3-N}{3} \text{ multiplications et additions}\end{aligned}$$

d'où au total $\dfrac{4N^3+3N^2-7N}{6}$ opérations élémentaires auxquelles il faut ajouter les N^2 nécessaires à la remontée du système. Lorsque N est grand, les termes en N^2 et N sont négligeables devant N^3 et le nombre d'opérations est donc de l'ordre de $\dfrac{2N^3}{3}$.

Remarque 6.6 On montre que ce nombre d'opérations est pratiquement optimal pour la résolution directe d'un système linéaire quelconque (c'est-à-dire sans aucune particularité). C'est pourquoi la méthode de Gauss est souvent utilisée dans le cas des matrices "pleines".

Noter que dans le temps de calcul, il faut aussi tenir compte de la stratégie du pivot : celle-ci peut prendre un temps non négligeable dans le cas d'un pivot total.

Les formules de Cramer nécessitent le calcul de $(n+1)$ déterminants et n divisions. Chaque déterminant calculé selon sa définition requiert $(n-1)\,n!$ multiplications, $n!-1$ additions soit $\left(n^2-1\right)n!+(n+1)!-(n+1)$ opérations. Ainsi pour $n=10$ cela donne environ 400 000 000 opérations contre environ 900 opérations par la méthode de Gauss.

6.2.2 Dérivés de la méthode de Gauss

Nous allons voir dans ce paragraphe plusieurs variantes de la méthode de Gauss (Crout, Doolittle, Gauss-Jordan) qui auront chacune leur intérêt propre.

6.2.2.1 Factorisation LU

Supposons qu'on doive, dans un programme donné, résoudre plusieurs fois le même système linéaire

$$AX = b$$

avec différents seconds membres b, mais la même matrice A. Si tous les seconds membres b sont initialement connus, on peut alors effectuer simultanément sur tous les seconds membres les manipulations intervenant dans l'élimination de Gauss.

Le plus souvent, dans la pratique, il s'agit de résoudre des systèmes avec des b calculés au cours du processus et donc inconnus initialement. Il n'est alors pas raisonnable de

recommencer la triangulation de Gauss à chaque résolution : il faut conserver ce résultat qui consiste en fait à factoriser la matrice A sous la forme $A = LU$ ou L est une matrice triangulaire inférieure et U une matrice triangulaire supérieure (L pour "lower", U pour "upper") : on conservera en mémoire L et U. Par la suite, tout système

$$(S) \qquad AX = b \Longleftrightarrow LUX = b$$

sera remplacé par la résolutions des deux systèmes équivalents

$$\left\{ \begin{array}{c} LY = b \\ UX = Y \end{array} \right. .$$

Il s'agit alors de systèmes triangulaires qui se résolvent par une méthode de "descente" puis de "remontée", soit un nombre d'opérations égal à $2N^2$.

Montrons comment l'élimination de Gauss donne une factorisation de A sous la forme $A = LU$, tout au moins quand aucune stratégie du pivot n'est appliquée ce que nous supposons dans ce qui suit. Notons $l_{ip} = a^p_{ip}/a^p_{pp}$. La matrice triangulaire obtenue en fin de méthode de Gauss est , avec les notations précédentes :

$$U = \left[\begin{array}{ccccccc} a^1_{11} & a^1_{12} & \cdots & \cdots & \cdots & \cdots & a^1_{1\mathrm{N}} \\ 0 & a^2_{22} & a^2_{23} & \cdots & \cdots & \cdots & a^2_{2\mathrm{N}} \\ \vdots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & a^p_{pp} & \cdots & a^p_{p\mathrm{N}} \\ \vdots & & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a^N_{\mathrm{NN}} \end{array} \right].$$

Si nous notons

$$L = \left[\begin{array}{cccc} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ l_{\mathrm{N}1} & \cdots & \cdots & l_{\mathrm{NN}} \end{array} \right] \text{ (noter } l_{ii} = 1)$$

on vérifie qu'on a alors

$$A = LU.$$

En effet le terme général du produit LU est par définition

$$a'_{ij} = \sum_{p=1}^{\min(i,j)} l_{ip} a^p_{pj}.$$

D'après les relations (6.3) définissant l'algorithme de Gauss, une sommation à i, j fixés de $p = 1$ à $\min(i,j) - 1$ donne

$$\sum_{p=1}^{\min(i,j)-1} a^{p+1}_{ij} = \sum_{p=1}^{\min(i,j)-1} a^p_{ij} - \sum_{p=1}^{\min(i,j)-1} l_{ip} a^p_{pj},$$

soit après réduction des termes identiques

$$\sum_{p=1}^{\min(i,j)-1} l_{ip}a_{pj}^{p} + a_{ij}^{\min(i,j)} = a_{ij}^{1}.$$

Puisque $a_{ij}^{1} = a_{ij}$, pour obtenir $a_{ij}' = a_{ij}$, il suffit de vérifier

$$l_{iq}a_{qj}^{q} = a_{ij}^{q} \text{ si } q = \min(i,j). \tag{6.8}$$

Si $i \leq j$ et donc $q = i$, c'est immédiat puisque $l_{ii} = 1$. Si $i > j$, soit $q = j$, on a par définition de l_{ij}

$$l_{ij} = \frac{a_{ij}^{j}}{a_{jj}^{j}}$$

...ce qui est (6.8).

Remarque 6.7 nous avons ainsi démontré que, si la triangulation de Gauss peut-être conduite sans stratégie du pivot, alors la matrice A admet une factorisation LU. On peut trouver des conditions suffisantes simples pour que ceci soit possible : par exemple lorsque A est symétrique définie positive. Ceci sera explicité au paragraphe suivant (factorisation de Cholesky).

Si A est inversible, nous avons vu que l'élimination de Gauss est toujours possible pourvu qu'on applique une stratégie du pivot partiel qui revient à permuter des lignes de A. Ceci prouve que si A est inversible, il existe une matrice P produit de matrices de transposition telle que PA ait une factorisation LU. Dans la pratique, une fois les permutations nécessaires repérées on les effectue préalablement à la résolution de tout nouveau système associé à A.

6.2.2.2 Algorithme de Crout

C'est le nom donné dans la littérature à une variante de l'algorithme de Gauss : elle suppose a priori l'existence de la décomposition $A = LU$, les coefficients l_{ij}, u_{ij} étant déterminés par le système d'équations

$$\left\{ \sum_{p=1}^{\min(i,j)} l_{ip}u_{pj} = a_{ij} \right\} \begin{array}{l} 1 \leq i \leq N \\ 1 \leq j \leq N \end{array}.$$

Ceci est un système de N^2 équations à $N^2 + N$ inconnues. On peut se fixer N éléments par exemple

$$l_{ii} = 1 \quad \forall i = 1, ..., N \quad \text{ou} \quad u_{ij} = 1 \quad \forall j = 1, ..., N.$$

Le système peut être alors résolu de proche en proche. Par exemple, avec le choix de $l_{ii} = 1$, on est conduit à :

$$\left\{ \begin{array}{l} l_{ii} = 1, i = 1, ..., N \\ \text{pour } r = 1, ..., N \\ \left\{ \begin{array}{l} u_{rj} = a_{rj} - \sum_{k=1}^{r-1} l_{rk}u_{kj} \quad j = r, ..., N \\ l_{ir} = \dfrac{a_{ir} - \sum_{k=1}^{r-1} l_{ik}u_{kr}}{u_{rr}} \quad i = r+1, ..., N \end{array} \right. \end{array} \right. . \tag{6.9}$$

Ceci détermine la matrice cherchée

$$M = \begin{bmatrix} u_{11} & \cdots & \cdots & u_{1N} \\ l_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ l_{N1} & \cdots & \cdots & u_{NN} \end{bmatrix} \quad \text{dans l'ordre}$$

Remarque 6.8 Si les éléments a_{ij} ne sont pas nécessaires ultérieurement, on peut présenter l'algorithme 6.9 sous forme plus "compacte", en écrasant les éléments de A par ceux de M au fur et à mesure qu'ils ne sont plus utilisés, d'où :

Algorithme de Crout compact :

$$\left[\begin{array}{l} \text{De } i = 2 \text{ à } N \quad a_{i1} := \frac{a_{i1}}{a_{11}} \\ \left[\begin{array}{l} \text{De } r = 2 \text{ à } N \\ \left[\begin{array}{l} \text{de } j = r \text{ à } N \\ a_{rj} := a_{rj} - \sum_{k=1}^{r-1} a_{rk} a_{kj} \\ \text{De } i = r+1 \text{ à } N \\ \\ a_{ir} := \dfrac{a_{ir} - \sum_{k=1}^{r-1} a_{ik} a_{kr}}{a_{rr}} \end{array}\right. \end{array}\right. \end{array}\right. .$$

Remarque 6.9 Aucune stratégie du pivot n'étant appliquée, l'algorithme de Crout peut conduire à des u_{rr} nuls (ou petits) et donc à une méthode incorrecte. Il faut ainsi l'appliquer uniquement à des matrices dont la décomposition LU est assurée et pour lesquelles les $|u_{kk}|$ seront assez grands.

- Les algorithmes de Gauss et de Crout ne diffèrent que par l'ordre des opérations : ils sont par ailleurs équivalents comme on le vérifie à l'aide des formules (6.3) et (6.9) : "$u_{rj} = a_{rj} - \sum_{k=1}^{r-1} l_{rk} u_{kj}$" n'est pas autre chose que a_{rj}^r dans la notation (6.3). Dans la méthode de Gauss, on effectue et on mémorise individuellement les produits $l_{rk} u_{kj}$; dans la méthode de Crout, chaque produit scalaire est traité comme un tout ce qui peut être plus avantageux lorsqu'on dispose d'un calculateur accumulant les produits scalaires en double précision.
- Dans l'algorithme de Crout, le déterminant est obtenu par

$$\det A = u_{11} u_{22} ... u_{NN}.$$

6.2.2.3 Cas particulier des matrices tridiagonales

Théorème 6.1 *Soit*

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & c_{N-1} \\ 0 & \cdots & \cdots & \cdots & 0 & a_N & b_N \end{bmatrix}$$

une matrice tridiagonale. On note A_k la matrice extraite de A en ne prenant que les k premières lignes et premières colonnes et on suppose $\det A_k \neq 0, \quad \forall k = 1, ..., N$. *Alors A admet la décomposition*

$$A = \begin{bmatrix} \frac{c_1}{z_1} & & & & \\ a_2 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & a_N & \frac{c_N}{z_N} \end{bmatrix} \begin{bmatrix} 1 & z_1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & z_{N-1} \\ & & & & 1 \end{bmatrix} = LU$$

où

$$z_1 = \frac{c_1}{b_1}, \quad z_k = \frac{c_k}{b_k - a_k z_{k-1}}, \quad k = 2, 3, ..., N \quad (c_N = 1 \text{ } \textit{par exemple}) \tag{6.10}$$

Le système $AX = b$ se résoud alors avec l'algorithme suivant :

$$w_1 = \frac{d_1}{b_1}, \quad w_k = \frac{d_k - a_k w_{k-1}}{b_k - a_k z_{k-1}}, \quad k = 2, 3, ..., N \text{ (descente)} \tag{6.11}$$

$$x_N = w_N, \quad x_k = w_k - z_k x_{k+1}, \quad k = N-1, N-2, ..., 1 \text{ (remontée)} \tag{6.12}$$

Remarque 6.10 Cette méthode de résolution requiert $3\,(N-1)$ additions (on ne calcule qu'une fois $b_k - a_k z_{k-1}$)
$3\,(N-1)$ multiplications
$2N$ divisions
soit $8N-6$ opérations au total, ce qui constitue une réduction considérable par rapport au cas d'une matrice pleine. Cette méthode directe est fortement conseillée pour la résolution d'un système tridiagonal.

Démonstration du Théorème 6.1 : On vérifie directement que le produit des deux matrices ci-dessus redonne A ; la seule chose à prouver est que les nombres z_k sont bien définis. Or, on montre par récurrence que

$$z_k = c_k \frac{\det A_{k-1}}{\det A_k}, \quad k = 2, ..., N \text{ (par hypothèse } \det A_k \neq 0)\,.$$

En effet, développant $\det A_k$ suivant la dernière ligne, on vérifie que :

$$\det A_k = b_k \det A_{k-1} - a_k c_{k-1} \det A_{k-2}.$$

Ainsi, si $z_{k-1} = c_{k-1} \frac{\det A_{k-2}}{\det A_{k-1}}$, d'après la définition (6.10) de z_k, on a :

$$z_k = \frac{c_k}{b_k - a_k c_{k-1} \frac{\det A_{k-2}}{\det A_{k-1}}} = \frac{c_k \det A_{k-1}}{\det A_k} \text{ d'après (6.11).}$$

La relation se montre directement pour $k = 2$.
Le système (6.11) correspond à la résolution de $LW = d$; le système (6.12) correspond à celle de $UX = W$.

6.2.2.4 Méthode de Gauss-Jordan

Autre variante de la méthode de Gauss : méthode de Gauss-Jordan : au lieu de déterminer M pour que MA soit triangulaire, on détermine M pour que MA soit diagonale : la résolution du système final est alors plus simple. Cependant, la diagonalisation demande plus de calculs et globalement le nombre d'opérations est sensiblement le même que dans la résolution de Gauss. Pour des raisons d'organisation des calculs, on emploie souvent la technique de Gauss-Jordan pour calculer l'inverse d'une matrice. Comme indiqué précédemment, on résoud alors simultanément les n systèmes linéaires

$$\{AX_i = e_i \ , i = 1, ..., N\}$$

où $e_i = (0, ..., 0, 1, 0, ..., 0)$ (le 1 est sur la ième coordonnée) et X_i est alors le i-ème vecteur colonne de A^{-1}.
Décrivons le passage de l'état p à l'état $p+1$: la matrice A_p a la structure suivante :

$$A_p = \begin{bmatrix} a_{11}^p & 0 & \cdots & 0 & a_{1p}^p & \cdots & a_{1\mathrm{N}}^p \\ 0 & a_{22}^p & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots & & \vdots \\ \vdots & & & 0 & a_{pp}^p & \cdots & a_{p\mathrm{N}}^p \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & a_{\mathrm{N}p}^p & \cdots & a_{\mathrm{NN}}^p \end{bmatrix}$$

On utilise alors la p-ième ligne (c'est-à-dire la p-ième équation) pour faire apparaître des zéros dans toute la colonne de rang p, sauf en a_{pp}^p (c'est-à-dire éliminer l'inconnue x_p dans toutes les autres équations). Comme dans la méthode de Gauss, on utilise donc a_{pp}^p comme pivot (après application éventuelle d'une stratégie), et si $l_{ip} = a_{ip}^p / a_{pp}^p$, A_{p+1} est obtenu par :

$$\begin{cases} a_{ij}^{p+1} = a_{ij}^p - l_{ip} a_{pj}^p \\ i = 1, ..., N, \ \ i \neq p \\ \quad p+1 \leq j \leq N \end{cases} . \tag{6.13}$$

On fait bien sûr les mêmes transformations aux seconds membres que, de manière pratique, on assimile à des colonnes supplémentaires de A ; les formules (6.13) sont alors appliquées à la matrice augmentée (j varie de $p+1$ à $N+k$, si k est le nombre de seconds membres)

6.2.3 Factorisation de Cholesky

Ce paragraphe traite du cas particulier des matrices symétriques définies positives. Pour de telles matrices, la méthode de Gauss sans stratégie du pivot s'applique toujours. En effet, on remarque d'abord que les sous-matrices principales $A_{(k)}$ sont inversibles :

$$A = \left[\begin{array}{c|c} A_{(k)} & \\ \hline & \end{array}\right].$$

Pour cela, soit $v \in \mathbb{R}^k$ tel que $A_{(k)}v = 0$; soit $\widetilde{v} \in \mathbb{R}^N$ défini par

$$\widetilde{v_i} = v_i, \;\; 1 \leq i \leq k \qquad \widetilde{v_i} = 0, \;\; k+1 \leq i \leq N.$$

On a

$${}^t v A_{(k)} v = {}^t \widetilde{v} A v = 0 \Longrightarrow v = 0,$$

puisque A est définie positive.
Maintenant, la condition $\det A_{(k)} \neq 0 \;\; \forall k$ est suffisante pour assurer que les pivots sont non nuls dans la factorisation de Gauss. En effet, d'après les règles de calcul des déterminants, si A^p est la matrice à l'étape p, on a :

$$\det A^p_{(p)} = \det A_{(p)}$$

où $A^p_{(p)}$ est la sous-matrice principale de A^p d'ordre p.

$$A^p_{(p)} = \begin{bmatrix} a^p_{11} & 0 & \cdots & 0 & a^p_{1p} \\ 0 & a^p_{22} & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \\ & & \ddots & \ddots & \vdots \\ 0 & & \cdots & 0 & a^p_{pp} \end{bmatrix}$$

Or $\det A^p_{(p)} = a^p_{11} ... a^p_{pp}$. Ce qui montre que $a^p_{pp} \neq 0$. Par récurrence, on voit donc qu'on est toujours conduit à des pivots non nuls. On en déduit :

Théorème 6.2 (Factorisation de Choleski) *Soit A une matrice symétrique définie positive. Alors il existe B triangulaire inférieure telle que :*

$$A = B\, B^t.$$

Remarque 6.11 On obtient bien sûr une décomposition de type LU, mais il n'y a qu'une matrice à déterminer. Cette matrice est parfois appelée racine carrée de A. On peut imposer que ses éléments diagonaux soient strictement positifs. La factorisation est alors unique (le montrer).
Dans la pratique, on utilise un algorithme d'identification (dit de Cholesky) pour calculer les éléments de B, plutôt que la méthode de Gauss. Celle-ci nous permet cependant de montrer l'existence de B.

Démonstration du Théorème 6.2 :
Comme montré ci-dessus, l'algorithme de Gauss sans stratégie de pivot peut être mené à terme et donne donc une factorisation de A en $A = LU$, soit

$$\begin{bmatrix} 1 & & & & & & \\ l_{21} & 1 & & & & & \\ \vdots & \ddots & \ddots & & & & \\ \vdots & & \ddots & \ddots & & & \\ \vdots & & & \ddots & \ddots & & \\ \vdots & & & & \ddots & \ddots & \\ l_{N1} & \cdots & \cdots & \cdots & l_{N,N-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \ddots & & & & & \\ \vdots & \ddots & \ddots & & & & \\ \vdots & & \ddots & \ddots & & & \\ \vdots & & & \ddots & u_{kk} & \cdots & \cdots \\ \vdots & & & & \ddots & \ddots & \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & u_{NN} \end{bmatrix}.$$

Les u_{kk} sont strictement positifs car

$$u_{11}...u_{kk} = \det A_k > 0,$$

ce dernier point résultant du fait que A_k est symétrique définie posisive. On intercale la matrice diagonale $\Lambda = diag\left(\sqrt{u_{ii}}\right)$, soit :

$$A = (L\Lambda)\left(\Lambda^{-1}U\right) = BC.$$

Comme $A = {}^tA,\ BC = {}^tC{}^tB \Longrightarrow C\left({}^tB\right)^{-1} = B^{-1t}C.$
Puisque $C\left({}^tB\right)^{-1}$ est triangulaire supérieure et $B^{-1t}C$ triangulaire inférieure, l'égalité prouve qu'elles sont diagonales. Comme les éléments diagonaux sont égaux à 1, ceci prouve $B^{-1t}C = I$ ou $C = {}^tB$.

Méthode pratique de calcul de B : algorithme de Cholesky
On pose a priori

$$B = \begin{bmatrix} b_{11} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ b_{N1} & \cdots & \cdots & b_{NN} \end{bmatrix}$$

et on détermine $\{b_{ij}\}$ à l'aide du système d'équations

$$\left\{ \sum_{k=1}^{i} b_{ik}b_{jk} = a_{ij},\ \ 1 \le i \le j \le N \right\} \tag{6.14}$$

Il y a $\frac{N(N+1)}{2}$ inconnues pour $\frac{N(N+1)}{2}$ équations. La résolution peut se faire de proche en proche, colonne par colonne, comme suit :

Algorithme de Cholesky

$$\begin{bmatrix} \text{De } i = 1 \text{ à } N \\ b_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} b_{ik}^2} \\ \begin{bmatrix} \text{De } j = i+1 \text{ à } N \\ b_{ji} = \left(a_{ij} - \sum_{k=1}^{i-1} b_{ik}b_{jk}\right)/b_{ii} \end{bmatrix} \end{bmatrix}. \tag{6.15}$$

Remarque 6.12 – A priori il n'est pas clair que le calcul des racines carrées ci-dessus soit possible dans l'ensemble des réels : cependant, comme nous avons montré directement l'existence de B, nous savons que le système (6.14) admet une solution et qu'ainsi le calcul sera toujours possible. On obtient au passage l'unicité de la solution.
– Le déterminant de A est obtenu comme sous-produit par

$$\det A = b_{11}^2 b_{22}^2 ... b_{\mathrm{NN}}^2$$

puisque $\det A = (\det B)^2$.
– Une évaluation du nombre d'opérations conduit à
$\frac{N^3}{6}$ additions, $\frac{N^3}{6}$ multiplications, $\frac{N^2}{2}$ divisions, N extractions de racines carrées. Ceci se compare très avantageusement à la méthode de Gauss donnant $\frac{N^3}{3}$ additions, $\frac{N^3}{3}$ multiplications, $\frac{N^2}{2}$ divisions.

La méthode de Cholesky est donc fortement conseillée (plutôt que Gauss) quand la matrice du système est symétrique, définie, positive. Elle a aussi l'avantage de se révéler plus stable par rapport aux erreurs d'arrondi. Noter également l'estimation a priori

$$|b_{ik}| < \sqrt{a_{ii}} \ \ \forall i = 1, ..., N, \ \ \forall k = 1, ..., i$$

provenant de la relation

$$\sum_{k=1}^{i} b_{ik}^2 = a_{ii}$$

et qui assure que les b_{ik} ne deviendront jamais trop grands.

6.2.4 Autres méthodes directes

– Une autre méthode, connue sous le nom de méthode de Householder, consiste à multiplier la matrice A par des matrices de symétries orthogonale pour aboutir à une factorisation de la forme

$$A = QR$$

où R est triangulaire supérieure et Q une matrice orthogonale. Cette technique nécessite environ deux fois plus d'opérations que celle de Gauss mais le conditionnement du système initial ne peut qu'être amélioré. Cette factorisation constitue le fondement d'un des plus importants algorithmes de recherche des valeurs propres.
– Signalons aussi la méthode du gradient conjugué qui est à l'heure actuelle de plus en plus utilisée pour la résolution des systèmes creux. Elle repose sur la minimisation de la fonctionnelle quadratique $\Phi(x) = \frac{1}{2}{}^t XAX - bX$ et sera décrite dans le chapitre 9. c'est une méthode "semi-directe".

6.3 Exercices du chapitre 6

Exercice 6.1 Montrer que la factorisation $A = LU$ est unique si on impose que les éléments diagonaux de L soient tous égaux à 1 (et A inversible).

Exercice 6.2 Montrer que la structure-bande est conservée dans la factorisation.

Exercice 6.3 Montrer qu'une permutation de lignes de A revient à multiplier A à gauche par une matrice T simple (on l'appelle matrice de transposition).

Exercice 6.4 Montrer que, dans l'algorithme de Crout compact $u_{kk} = \frac{\det A_k}{\det A_{k-1}}$ $(k \geq 2)$ où A_k est la matrice de rang k extraite de A :

$$A = \left[\begin{array}{c|c} A_k & \\ \hline & \end{array}\right]$$

6.4 TD6 : résolution des systèmes linéaires

Exercice 1 : Instabilités. On considère la matrice de Hilbert $\mathcal{H}$ de taille n dont les éléments sont donnés par $\mathcal{H}_{ij} = 1/(i+j-1)$, pour $1 \leq i, j \leq n$. Soit $\mathbf{x}_{ref}$ le vecteur dont toutes les composantes sont égales à 1 et définissons un vecteur second membre $\mathbf{b} = \mathcal{H}\mathbf{x}_{ref}$. On définit $\mathbf{x}_{num}$ comme la solution numérique par la méthode de Gauss du système : $\mathcal{H}\mathbf{x}_{num} = \mathbf{b}$. Tracer l'erreur relative (en norme euclidienne) entre $\mathbf{x}_{ref}$ et $\mathbf{x}_{num}$ en fonction de n. Que se passe t-il ? Comment expliquez-vous ce comportement ?

Exercice 2 : Résolution d'une EDP par différences finies. Le but de cet exercice est la comparaison de différentes méthodes pour la résolution d'un système linéaire obtenu par discrétisation d'un problème de Dirichlet.

Soit $\Omega =]0,1[\times]0,1[$ le carré unité, considérons le problème suivant :

$$\begin{cases} -\Delta u = f(x,y) = 13\pi^2 \sin(3\pi x)\sin(2\pi y) \ \text{ dans } \ \Omega \\ u = 0 \ \text{ sur } \ \partial\Omega \end{cases}$$

Pour résoudre ce problème par différences finies, utilisons le maillage formé des n^2 points P_k de coordonnées $(x_i, y_j) = (\frac{i}{n+1}, \frac{j}{n+1})$ avec $1 \leq i \leq n$ et $1 \leq j \leq n$ (Les points sont numérotés de gauche à droite puis de bas en haut). La discrétisation consiste à remplacer le laplacien par la formule centrée points équidistants d'ordre 2 :

$$-\Delta u(x,y) \approx \frac{1}{h^2}(4u(x,y) - u(x+h,y) - u(x-h,y) - u(x,y+h) - u(x,y-h))$$

L'inconnue du problème est alors le vecteur $u = (u_k)$, à n^2 composantes qui sont les valeurs approchées de la solution aux points P_k, u vérifie le système linéaire suivant :

$$Au = h^2 b$$

avec :
h pas du maillage,
A matrice de discrétisation, par exemple pour $n = 3$, A s'écrit :

$$A = \begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

b second menbre de l'équation $b = (b_k)$ avec $b_k = f(P_k)$.

1. Ecrire une fonction qui étant donné un entier n renvoie la matrice A correspondante.
Fonctions Matlab utiles : diag, repmat

2. Ecrire une fonction qui étant donné un entier n renvoie les deux vecteurs X et Y respectivement abscisses et ordonnées des points (P_k). Définir alors le vecteur b second membre de l'équation.

3. Résoudre le système linéaire par les trois méthodes suivantes : pivot de Gauss, factorisation LU, factorisation de Cholesky. On comparera notamment l'erreur entre la solution obtenue et la solution exacte du problème : $u_{th}(x, y) = \sin(3\pi x)\sin(2\pi y)$ définie par :

$$erreur = \frac{\sum_k |u(P_k) - u_{th}(P_k)|^2}{\sum_k |u_{th}(P_k)|^2}$$

Et également le temps de calcul nécessaire et ce pour différentes valeurs de n.
Fontions Matlab utiles : \, lu, chol, tic et toc

4. (si vous avez le temps) Quelle place mémoire est nécessaire pour stocker la matrice A correspondant à $n = 40$? La plupart des élements de A sont pourtant nuls ! Il existe une méthode de stockage de ces matrice dites creuses consistant à ne garder en mémoire que les élements non nuls de A et leurs positions dans la matrice. Utiliser ce genre de stockage pour le matrice A. Quelle est la nouvelle place mémoire utilisée ?
Comparer à nouveau le temps de calcul des méthode de Gauss, de factorisation LU, de factorisation de Choleski dans le cas où A est stockée sous forme de matrice creuse. Quelle est votre conclusion ? Quel est l'inconvénient des méthodes de factorisation ?
Fontions Matlab utiles : whos sparse spdiags

6.5 Aide-mémoire : Rappels sur les matrices

Rappelons quelques définitions et notations pour les matrices et le calcul matriciel. Une matrice A de taille $n \times m$ est un tableau de nombre avec n lignes et m colonnes. Le terme situé sur la i-ème ligne et la j-ième colonne sera noté $a_{i,j}$. On dit que la matrice est carrée lorsque $m = n$. Un vecteur peut être regardé comme une matrice $n \times 1$ (vecteur colonne) ou $1 \times n$ (vecteur ligne). Pour faire le lien avec le cours de mécanique : les vecteurs correspondent aux tenseurs d'ordre 1, tandis que les matrices dont il est question ici correspondent aux tenseurs d'ordre 2.

On peut ajouter entre elles deux matrices de même dimension, le terme général de $A + B$ est alors naturellement $a_{i,j} + b_{i,j}$. L'ensemble des matrices de taille $n \times m$ forme un espace vectoriel (le produit de A par le scalaire k est la matrice de terme général $ka_{i,j}$). Il est de dimension nm et une base, appelée canonique, est constituée des matrices $E_{k,l}, 1 \leq k \leq n$, $1 \leq l \leq m$ dont le terme général est $\delta_{i,k}\delta_{j,l}$ qui vaut 1 pour $i = k$ et $j = l$ et 0 sinon.

On ne peut faire le produit matriciel $A \times B$ (ou plus simplement AB que si le nombre de colonnes de A est égal au nombre de lignes de B. Ainsi le produit de A de taille $n \times p$ par B de taille $p \times m$ donnera une matrice C de taille $n \times m$ dont le terme général est

$$c_{i,j} = \sum_{k=1}^{p} a_{i,k}\, b_{k,j}\,.$$

Un cas particulier est le produit de la matrice A de taille $n \times m$ par le vecteur colonne X de taille $m \times 1$: cela donne donc un vecteur colonne de taille $n \times 1$. Le produit de deux matrices carrées est donc toujours possible, mais il n'est **pas commutatif** : en général $A \times B \neq B \times A$.

On appelle **transposée de la matrice** A, la matrice obtenue en échangeant les lignes et les colonnes de A. On la notera A^t (ou A' en Matlab) et son terme général est donc $a_{j,i}$.
Une matrice carrée A est dite symétrique si $A = A^t$ autrement dit si $a_{i,j} = a_{j,i}$ pour tout i et j.
Une matrice A symétrique est dite positive si $X^t\, A\, X \geq 0$ pour tout $X \in \mathbb{R}^n$. Notons que $X^t\, A\, X$ est un réel.
Une matrice A symétrique est dite définie positive si $X^t\, A\, X > 0$ pour tout $X \in \mathbb{R}^n$ non nul.

On appelle matrice identité, et on note Id, la matrice carrée possédant des 1 sur sa diagonale et des 0 partout ailleurs. C'est l'élément neutre pour la multiplication au sens où $A\, Id = Id\, A = A$ pour toute matrice A.
On dit qu'une matrice carrée A est inversible, s'il existe une matrice carrée, notée A^{-1} telle que $A\, A^{-1} = A^{-1}\, A = Id$. Un critère nécessaire et suffisant d'inversibilité est que le déterminant de A soit non nul.

Chapitre 7

Résolution de systèmes linéaires - méthodes itératives

7.1 Analyse du conditionnement d'un système linéaire

Le but de ce paragraphe est d'étudier l'influence d'une petite variation du second membre ou des coefficients d'une matrice d'un système linéaire sur la valeur de la solution. Comme il a déjà été expliqué dans le premier chapitre de ce cours, les arrondis faits sur les données (et inévitables dans un calcul numérique sur machine), peuvent être interprétés comme de telles "petites variations" : il est donc essentiel de connaitre leur impact sur le résultat calculé.

7.1.1 Quelques préliminaires sur les normes matricielles

On rappelle tout d'abord :

Définition 7.1 *Etant donné V un espace vectoriel sur $\mathbb{R}$ ou $\mathbb{C}$, on appelle norme sur V toute application (qu'on notera $\|.\|$) de V dans $[0,\infty[$ telle que*

(i) $\|x\| = 0 \Longleftrightarrow x = 0$ *pour* $x \in V$

(ii) $\forall x \in V, \forall \lambda \in \mathbb{R}$ (*ou* $\mathbb{C}$), $\quad \|\lambda x\| = |\lambda| \, \|x\|$ *(homogénéité)*

(iii) $\forall x, y \in V, \quad \|x+y\| \leq \|x\| + \|y\|$ *(inégalité triangulaire).*

Nous serons, en particulier, amenés à utiliser les normes classiques de $\mathbb{R}^N$ (ou $\mathbb{C}^N$) à savoir :

$$\begin{aligned} \|x\|_2 &= \left(\sum_{i=1}^{N} x_i^2 \right)^{1/2} \\ \|x\|_1 &= \sum_{i=1}^{N} |x_i| \\ \|x\|_\infty &= \max_{1 \leq i \leq N} |x_i| \, . \end{aligned}$$

Plus généralement, on peut considérer

$$\|x\|_p = \left(\sum_{i=1}^{N} |x_i|^p \right)^{1/p}, \quad 1 \leq p < \infty.$$

La notation $\|x\|_\infty$ utilisée ci-dessus provient du fait que

$$\forall x \in \mathbb{R}^N, \lim_{p \to \infty} \|x\|_p = \max_{1 \leq i \leq N} |x_i| \quad (= \|x\|_\infty).$$

Il est bon de connaitre la géométrie des boules-unités pour chacune de ces normes soit $B = \left\{ x \in \mathbb{R}^N ; \|x\| \leq 1 \right\}$. Ainsi, en dimension 2, on a :

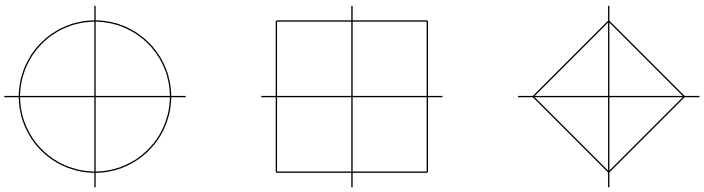

FIGURE 7.1 – Les boules unités, pour la norme $\|.\|_2$ (à gauche), pour la norme $\|.\|_\infty$ (au centre), pour la norme $\|.\|_1$ (à droite)

On remarque que la boule-unité pour la norme euclidienne est "arrondie" : elle ne présente ni angle, ni partie plate contrairement aux deux autres. Cette géométrie particulière explique partiellement pourquoi la norme euclidienne a de "meilleurs" comportements que les autres.

7.1.1.1 Norme sur l'espace vectoriel des matrices

Une matrice carrée d'ordre N est la donnée de N^2 réels (ou complexes) : elle peut donc être identifiée à un élément de $\mathbb{R}^{N^2}$. A ce titre, toute norme sur $\mathbb{R}^{N^2}$ fournit une norme sur l'espace des matrices. Ainsi, si $A = [a_{ij}]$, on peut poser

$$\|A\| = \max_{1 \leq i,j \leq N} |a_{ij}| \text{ ou } \|A\| = \left(\sum_{i,j} a_{ij}^2 \right)^{1/2}, \text{ etc...}$$

Cependant, pour qu'une norme soit pratique d'utilisation, il faut aussi qu'elle soit compatible avec le produit des matrices, ce qui nous conduit à exiger une quatrième propriété :

$$(iv) \qquad \|AB\| \leq \|A\| \, \|B\|.$$

Définition 7.2 *On appelle norme matricielle toute norme sur l'espace vectoriel des matrices vérifiant* (iv).

Etant donnée une norme vectorielle $\|.\|$ définie sur $\mathbb{R}^N$, il existe une norme matricielle naturellement associée à $\|.\|$: on l'appelle norme matricielle induite par $\|.\|$ et elle est définie par :

$$\|A\| = \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|} \left(= \max_{\|x\|=1} \|Ax\| \right).$$

Remarque 7.1 – L'égalité entre ces deux expressions est due à la linéarité de A soit $\frac{Ax}{\|x\|} = A\left(\frac{x}{\|x\|}\right)$ et $\left\|\frac{x}{\|x\|}\right\| = 1$. La deuxième expression permet de montrer l'existence du maximum en remarquant que $x \to \|Ax\|$ est continue sur la sphère unité de $\mathbb{R}^N$ qui est compacte.

– On vérifie aisément que l'application $A \to \|A\|$ est bien une norme matricielle. Vérifions par exemple (iv) :

$$\|AB\| = \max_{x\neq 0} \frac{\|ABx\|}{\|x\|}$$

Puisque $\|ABx\| \leq \|A\| \|Bx\|$ par définition de $\|A\|$, on a donc

$$\|AB\| \leq \max_{x\neq 0} \|A\| \frac{\|Bx\|}{\|x\|} \leq \|A\| \|B\|.$$

– On garde souvent la même notation pour la norme vectorielle et la norme matricielle induite bien que les deux normes n'agissent pas sur le même espace : cela ne conduit en fait à aucune confusion et favorise au contraire les écritures. Ainsi, on écrira :

$$\|AX\| \leq \|A\| \|X\|, \forall X \in \mathbb{R}^N, \forall A \text{ matrice carrée d'ordre } N.$$

– On notera de la même façon $\|.\|_p$ la norme matricielle associée à $\|.\|_p$.

Théorème 7.1

$$\begin{aligned}
&(i)\ \|A\|_\infty &&= \max_{1\leq i\leq N} \left(\sum_{j=1}^{N} |a_{ij}| \right) \\
&(ii)\ \|A\|_1 &&= \max_{1\leq j\leq N} \left(\sum_{i=1}^{N} |a_{ij}| \right) \\
&(iii)\ \|A\|_2 &&= \sqrt{\rho(A^*A)} \text{ où } A^* = \overline{A}^t \text{ (adjointe de } A)
\end{aligned}$$

et $\rho(M)$ désigne le rayon spectral de la matrice M soit

$$\rho(M) := \max\{|\lambda| ; \lambda \text{ valeur propre de } M\}.$$

Remarquons que la matrice adjointe A^* n'est autre que la transposée dans le cas d'une matrice à coefficients réels. Dans le cas particulier où A est symétrique, $\|A\|_2 = \rho(A)$.

Remarque 7.2 – Toute racine carrée d'une valeur propre de A^*A est appelée valeur singulière de A. Si A est normale (i.e. $AA^* = A^*A$), alors $\sqrt{\rho(A^*A)} = \rho(A)$. Dans tous les cas $\rho(A^*A) = \rho(AA^*)$.

– $\|A\|_\infty$ et $\|A\|_1$ se calculent simplement à partir de la matrice de A contrairement à $\|A\|_2$.

– On utilise aussi la norme, dite de Fröbenius (ou norme de Schur) :

$$\|A\|_F = \left(\sum_{i,j} a_{ij}^2 \right)^{1/2} = \left[trace\,(A^*A) \right]^{1/2} .$$

On montre qu'il s'agit d'une norme matricielle, mais non induite par une norme vectorielle (en effet, si I est l'identité, $\|I\|_F = \sqrt{N}$ ... au lieu de 1 pour une norme induite par une norme vectorielle).

Démonstration du Théorème 7.1 :

$$\begin{aligned} \|Ax\|_\infty &= \max_{1\le i\le N} \left\{ \left| \sum_{j=1}^N a_{ij} x_j \right| \right\} \le \max_{1\le i\le N} \left\{ \sum_{j=1}^N |a_{ij}|\,|x_j| \right\} \\ &\le \max_{1\le i\le N} \left\{ \left(\sum_{j=1}^N |a_{ij}| \right) \left(\max_{1\le j\le N} |x_j| \right) \right\} \le \|x\|_\infty \max_{1\le i\le N} \left(\sum_{j=1} |a_{ij}| \right) . \end{aligned}$$

Ceci démontre que $\|A\| \le \max_{1\le i\le N} \left(\sum_{j=1}^N |a_{ij}| \right)$. On montre que l'égalité est réalisée en considérant le vecteur $x = (\varepsilon_j)$ où

$$\varepsilon_j = \begin{cases} 1 \text{ si } a_{i_0 j} \ge 0 \\ -1 \text{ sinon } a_{i_0 j} < 0 \end{cases} \quad \text{et } i_0 \text{ est un indice tel que}$$

$$\sum_{j=1}^N |a_{i_0 j}| = \max_{1\le i\le N} \left\{ \sum_{j=1}^N |a_{ij}| \right\}$$

- Le point (ii) s'obtient de façon analogue : noter la dualité entre les normes $\|.\|_1$ et $\|.\|_\infty$ qui va, en fait, bien au delà de cette simple remarque.
- On a $\|A\|_2^2 = \max_{\|v\|_2=1} \|Av\|_2^2 = \max_{\|v\|_2=1} v^*A^*Av$, puisque $\|x\|_2^2 = x^*x = \sum |x_i|^2$. La matrice A^*A est hermitienne puisque $(A^*A)^* = A^*A$.

On sait (cf. cours d'algèbre linéaire classique) qu'il existe une matrice unitaire U telle que $U^{-1}A^*AU = D$ soit diagonale, la diagonale étant formée des valeurs propres λ de A^*A. On a donc :

$$\max_{\|v\|_2=1} v^*A^*Av = \max_{\|v\|_2=1} v^*UDU^{-1}v.$$

Puisque U est unitaire, $\left\|U^{-1}v\right\|_2 = 1$. Puisque par ailleurs U^{-1} est bijective, on a donc :

$$\|A\|_2^2 = \max_{\|w\|_2=1} w^*Dw = \max_{\sum_{i=1}^N w_i^2=1} \left(\sum_{i=1}^N \lambda_i w_i^2 \right) = \max_{i=1,\dots,N} |\lambda_i| = \rho\,(D) = \rho\,(A^*A)\,.$$

7.1.2 Analyse du conditionnement

Considérons le système linéaire

$$AX = b. \tag{7.1}$$

Considérons une petite perturbation δb de b. La solution correspondante est perturbée soit

$$A\,(X + \delta X) = b + \delta b. \tag{7.2}$$

Nous allons mesurer la variation relative de X en fonction de celle de b. Pour cela, nous prenons $\|.\|$ une norme quelconque sur $\mathbb{R}^N$.
De (7.1), (7.2), on tire

$$A(\delta X) = \delta b \text{ soit } \delta X = A^{-1}(\delta b).$$

Utilisant la norme matricielle induite, on a :

$$\|\delta X\| \leq \left\|A^{-1}\right\| \|\delta b\|.$$

Par ailleurs, avec (7.1)

$$\|b\| \leq \|A\| \|X\|.$$

Ces deux inégalités donnent :

$$\frac{\|\delta X\|}{\|X\|} \leq \left\|A^{-1}\right\| \|A\| \frac{\|\delta b\|}{\|b\|}. \tag{7.3}$$

Cette estimation sur la variation de X est en fait optimale puisqu'on peut trouver X_0 tel que $\|AX_0\| = \|A\| \|X_0\|$ et δb tel que $\left\|A^{-1}\right\| \|\delta b\| = \left\|A^{-1}(\delta b)\right\|$.
Ainsi la variation relative sur X sera d'autant plus grande que le nombre $\left\|A^{-1}\right\| \|A\|$ est plus grand : on l'appelle le conditionnement de A relatif à la norme $\|.\|$.

Définition 7.3 *On appelle conditionnement de la matrice A (dans la norme matricielle $\|.\|$), le nombre*

$$cond(A) = \left\|A^{-1}\right\| \|A\|.$$

Si on travaille avec la norme $\|.\|_p$, on notera le conditionnement associé $cond_p$.

C'est le même nombre qui intervient lors de la variation des coefficients de A : supposons, en effet, que A soit perturbée par une matrice ΔA ; alors

$$\begin{aligned}
AX &= b \\
(A + \Delta A)(X + \delta X) &= b \\
&\Longrightarrow A(\delta X) + \Delta A(X + \delta X) = 0 \\
&\Longrightarrow \delta X = -A^{-1}\Delta A(X + \delta X) \\
&\Longrightarrow \|\delta X\| \leq \left\|A^{-1}\right\| \|\Delta A\| \|X + \delta X\| \\
&\Longrightarrow \frac{\|\delta X\|}{\|X + \delta X\|} \leq cond(A)\frac{\|\Delta A\|}{\|A\|}.
\end{aligned}$$

7.1.2.1 Propriétés de $cond(A)$

(i) $cond(A) \geq 1$ et le conditionnement est d'autant meilleur qu'il est plus proche de 1.
(en effet $AA^{-1} = I \Longrightarrow 1 = \|I\| \leq \|A\| \left\|A^{-1}\right\|$.)

(ii) $cond(A) = cond(A^{-1})$, $cond(\alpha A) = cond(A)$.

(iii) si A est normale (i.e. $A^*A = AA^*$), pour la norme $\|.\|_2$, on a

$$cond_2(A) = \frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}, \ \lambda_i \text{ valeur propre de } A.$$

Plus généralement, pour une matrice A quelconque

$$cond_2(A) = \frac{\mu_N(A)}{\mu_1(A)},$$

$$\begin{aligned} \mu_N &= \text{plus grande valeur singulière de } A, \\ \mu_1 &= \text{plus petite valeur singulière de } A. \end{aligned}$$

(iv) si A est unitaire (ie $U^* = U^{-1}$), $cond_2(A) = 1$.

Remarque 7.3 – La démonstration de (iii) est immédiate à partir de la définition du conditionnement, du Théorème 7.1 et des remarques qui le suivent.
- La propriété (iii) exprime qu'une matrice dont le spectre (i.e. l'ensemble des valeurs propres) est étendu sera mal conditionnée.
- La propriété (iv) justifie l'emploi de matrices unitaires dans certains procédés de factorisation (cf. méthode de Householder) : ainsi le conditionnement du système final est au moins aussi bon que celui du système initial.
- On ne modifie pas le conditionnement d'une matrice en la multipliant par un scalaire. Par contre, on peut diminuer $cond_2 A$ en multipliant certaines lignes ou certaines colonnes par des coefficients non nuls : il s'agit de l'équilibrage de la matrice. C'est une technique de préconditionnement très utilisée en pratique.

7.1.3 Exemple de système linéaire mal conditionné

(dû à R.S. Wilson, cf. P.G. Ciarlet "Introduction à l'analyse numérique matricielle et à l'optimisation")
Considérons le système

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix} \text{ de solution } \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Le système perturbé

$$\begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix} \begin{bmatrix} u_1 + \delta u_1 \\ u_2 + \delta u_2 \\ u_3 + \delta u_3 \\ u_4 + \delta u_4 \end{bmatrix} = \begin{bmatrix} 32,1 \\ 22,9 \\ 33,1 \\ 30,9 \end{bmatrix} \text{ a pour solution } \begin{bmatrix} 9,2 \\ -12,6 \\ 4,5 \\ -1,1 \end{bmatrix}.$$

Ainsi une erreur relative de 1/200 sur les données entraîne une erreur relative de l'ordre de 10/1 du résultat (erreur amplifiée de 2000).
De même

$$\begin{bmatrix} 10 & 7 & 8,1 & 7,2 \\ 7,08 & 5,04 & 6 & 5 \\ 8 & 5,98 & 9,89 & 9 \\ 6,99 & 4,99 & 9 & 9,98 \end{bmatrix} \begin{bmatrix} u_1 + \Delta u_1 \\ u_2 + \Delta u_2 \\ u_3 + \Delta u_3 \\ u_4 + \Delta u_4 \end{bmatrix} = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix} \text{ a pour solution } \begin{bmatrix} -81 \\ 137 \\ -34 \\ 22 \end{bmatrix}.$$

Pourtant, la matrice est "bonne" (symétrique, de déterminant 1, donc loin de 0). Son inverse est d'ailleurs donnée par

$$A^{-1} = \begin{bmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{bmatrix}.$$

Mais les valeurs propres de A sont

$$\lambda_1 \approx 0,01015 < \lambda_2 \approx 0,8431 < \lambda_3 \approx 3,858 < \lambda_4 \approx 30,2877, \text{ si bien que}$$

$$cond_2(A) = \frac{\lambda_4}{\lambda_1} \approx 2984 \text{ est grand !}$$

D'autre part

$$u = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \delta u = \begin{bmatrix} 8,2 \\ -13,6 \\ 3,5 \\ -2,1 \end{bmatrix}, b = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}, \delta b = \begin{bmatrix} 0,1 \\ -0,1 \\ 0,1 \\ -0,1 \end{bmatrix}$$

de sorte que

$$\frac{\|\delta u\|_2}{\|u\|_2} \simeq 8,1985 \text{ et } cond_2(A) \frac{\|\delta b\|_2}{\|b\|_2} \simeq 9,9424.$$

On n'est donc pas loin de l'égalité dans l'estimation (7.3)

$$\frac{\|\delta u\|_2}{\|u\|_2} \leq cond_2(A) \frac{\|\delta b\|_2}{\|b\|_2}.$$

7.1.4 Préconditionnement

Les matrices mal conditionnées sont véritablement à l'origine d'erreurs importantes dans les calculs pratiques, y compris pour les ingénieurs. Par exemple, dans la résolution d'une équation aux dérivées partielles avec une discrétisation de type différences finies ou éléments finis de pas h (très petit), il est classique que le conditionnement de la matrice du système linéaire soit en $O(\frac{1}{h^2})$! Par ailleurs pour les méthodes itératives comme la méthode du gradient conjugué (cf. chapitre 9), la rapidité de convergence est directement liée au conditionnement : plus celui-ci est mauvais, plus la convergence sera lente. Il est donc souvent utile de remplacer le système linéaire initial par un système équivalent *mieux conditionné*, c'est ce qu'on appelle la technique du préconditionnement. Il y a de nombreuses façons de le faire. Citons simplement ici sans détailler les méthodes de factorisation incomplète ou le préconditionnement SSOR.

7.2 Méthodes itératives

Etant donné le système $AX = B$ à résoudre, les méthodes itératives de résolution de systèmes linéaires consistent à calculer les valeurs successives d'une suite de vecteurs X^k convergeant vers la solution X quand $k \to \infty$.

Principe : On décompose A en $A = M - N$ où M est une matrice "facile" à inverser, au sens que le système $MY = d$ se résout facilement (M diagonale, ou triangulaire, ou diagonale par blocs...). Alors

$$AX = b \Longleftrightarrow MX = NX + b$$

conduit à l'itération

$$MX^{k+1} = NX^k + b.$$

7.2.1 Description des méthodes de Jacobi, Gauss-Seidel, relaxation

On suppose donnée A avec $a_{ii} \neq 0 \ \forall i$. On décompose A sous la forme

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & \cdots & a_{1\mathrm{N}} \\ a_{21} & a_{22} & & & -F & \\ & & & & & \\ & & & D & & \\ & -E & & & & \\ a_{\mathrm{N}1} & \cdots & \cdots & \cdots & a_{\mathrm{N},\mathrm{N}-1} & a_{\mathrm{NN}} \end{bmatrix} = D - E - F$$

où D est diagonale, E triangulaire inférieure, F triangulaire supérieure.
Une première décomposition conduit à

$$AX = b \Longleftrightarrow DX = (E + F)\, X + b$$

et à la méthode itérative de Jacobi :

$$\begin{cases} X^{k+1} = D^{-1}\,(E + F)\, X^k + D^{-1} b \\ X^0 \text{ arbitraire} \end{cases} \tag{7.4}$$

Algorithme de Jacobi

$$\left[\begin{array}{c} X^0 \text{ choisi} \\ \text{De } k = 0 \text{ à } \ldots \text{ (test d'arrêt)} \\ \left[\begin{array}{c} \text{De } i = 1 \text{ à } N \\ x_i^{k+1} := \left(- \sum_{p \neq i} a_{ip} x_p^k + b_i \right) / a_{ii}. \end{array} \right. \end{array} \right.$$

Remarque 7.4 Dans cet algorithme, il est nécessaire de conserver en mémoire tous les x_i^k aussi longtemps que le calcul des x_i^{k+1} n'est pas achevé. Il est possible d'utiliser 2 fois moins de mémoires en écrasant à chaque fois x_i^k par x_i^{k+1}, ce qui conduit à :

Algorithme de Gauss-Seidel

$$\left[\begin{array}{c} X^0 \text{ choisi} \\ \text{De } k = 0 \text{ à } \ldots \text{(test d'arrêt)} \\ \left[\begin{array}{c} \text{De } i = 1 \text{ à } N \\ x_i^{k+1} := \left(- \sum_{p<i} a_{ip} x_p^{k+1} - \sum_{p>i} a_{ip} x_p^k + b_i \right) / a_{ii}. \end{array} \right. \end{array} \right.$$

Remarque 7.5 Sous forme matricielle, cet algorithme s'écrit en fait :

$$DX^{k+1} = EX^{k+1} + FX^k + b$$

ou

$$X^{k+1} = (D-E)^{-1}\left(FX^k + b\right).$$

Cet algorithme peut encore être modifié en introduisant un paramètre supplémentaire qu'on choisira au mieux pour que la convergence soit plus rapide. Selon les notations traditionnelles, on est conduit à :

$$X^{k+1} = (1-\omega)\,X^k + \omega\left[(D-E)^{-1}\left(FX^k + b\right)\right].$$

On vérifie immédiatement que si X^k converge, il converge vers X solution de

$$X = (1-\omega)\,X + \omega\left[(D-E)^{-1}(FX+b)\right] \Longleftrightarrow X = (D-E)^{-1}(FX+b) \Longleftrightarrow AX = b.$$

Algorithme de relaxation

$$\left[\begin{array}{c} X^0 \text{ choisi} \\ \text{De } k=0 \text{ à } \ldots\text{(test d'arrêt)} \\ \left[\begin{array}{c} \text{De } i=1 \text{ à } N \\ x_i^{k+1} := (1-\omega)\,x_i^k - \omega\left(\sum_{p<i} a_{ip}x_p^{k+1} + \sum_{p>i} a_{ip}x_p^k - b_i\right)/a_{ii}. \end{array}\right. \end{array}\right.$$

Remarque 7.6 Il faut noter que chaque itération nécessite (dans Gauss-Seidel par exemple) :

$$N\left((N-1)\ \text{multiplications}\ +(N-1)\ \text{additions}\ +1\ \text{division}\right) \approx 2N^2\ \text{opérations}$$

si la matrice est pleine. Pour être comparable à la méthode de Gauss, il faut donc que la précision voulue soit obtenue en moins de $\frac{N}{3}$ itérations. Dans la pratique, on constate que les méthodes itératives sont surtout avantageuses lorsque la matrice est creuse : à chaque itération, le nombre d'opérations est d'ordre kN, où k est une constante fixe directement proportionnelle au nombre d'éléments non nuls. De plus, il suffit de stocker les éléments non nuls de A. C'est ce dernier point qui est essentiel dans le choix des méthodes itératives pour les grands systèmes creux : une factorisation de type Gauss ou Cholesky, même si elle préserve la structure-bande, remplit les "trous" à l'intérieur de la bande et nécessite donc plus d'espace mémoire.

7.2.2 Itérations par blocs

Supposons que la matrice A se présente naturellement sous la forme

$$A = \left[\begin{array}{ccccc} A_{11} & & & & A_{15} \\ & A_{22} & & & \\ & & A_{33} & & \\ & & & A_{44} & \\ A_{51} & & & & A_{55} \end{array}\right]$$

où les matrices A_{ii} sont inversibles. On peut alors utiliser un algorithme de relaxation par blocs : si $X^k = \left(X_1^k, X_2^k, X_3^k, X_4^k, X_5^k\right)$ où les X_i^k ont des longueurs adaptées à la décomposition ci-dessus, on a :

$$\left[\begin{array}{c} \text{Pour } i = 1, ..., 5 \\ A_{ii}X_i^{k+1} = (1-\omega)\, A_{ii}X_i^k - \omega\left[\sum_{p<i} A_{ip}X_p^{k+1} + \sum_{i<p<5} A_{ip}X_p^k - B_i\right] \end{array}\right. .$$

Ce regroupement par blocs peut accélérer la convergence, mais alourdit bien sûr le coût de chaque itération puisqu'il y a, à chaque pas, 5 sous-systèmes à résoudre. Ce n'est que si cette résolution est rapide que ce choix peut être intéressant.

7.2.3 Etude de la convergence des méthodes itératives

Considérons une méthode itérative du type

$$X^{k+1} = BX^k + d \tag{7.5}$$

appliquée à la résolution de $AX = b$. (On suppose A inversible).

Définition 7.4 *On dit que la méthode est consistante si, lorsque X^k converge vers Y, alors Y est la solution cherchée soit $Y = A^{-1}b$.*
On dit que la méthode est convergente si, pour tout choix X^0 de la donnée initiale, X^k converge vers $A^{-1}b$.

Remarque 7.7 Pour qu'une méthode soit consistante, il faut bien sûr choisir B et d en fonction de A et B convenablement : supposons que $X^k \to_{k\to\infty} Y$.
On a alors

$$\begin{aligned} Y &= BY + d \\ X &= A^{-1}b \\ Y - X &= B\,(Y - X) + d - (I - B)\, A^{-1}b. \end{aligned}$$

La consistance est assurée si $d = (I - B)\, A^{-1}b$ et $I - B$ est inversible.

Théorème 7.2 *Supposons la méthode (7.5) consistante. Alors elle est convergente si et seulement si*

$$\rho\,(B) < 1. \qquad (\rho\,(B) = \textit{rayon spectral de } B)\,. \tag{7.6}$$

Démonstration : Soit X la solution cherchée, i.e. (puisque la méthode est consistante) $X = BX + d$. Retranchant à (7.5), on obtient :

$$X^{k+1} - X = B\left(X^k - X\right)$$

et par récurrence

$$X^k - X = B^k\left(X^0 - X\right). \tag{7.7}$$

Puisque $X^0 - X$ est un vecteur arbitraire, on voit que $X^k - X$ tend vers 0 (pour tout choix de X^0) si et seulement si

$$\lim_{k\to\infty} B^k = 0 \tag{7.8}$$

(au sens que tous les éléments de la matrice B^k tendent vers 0).
Supposons que $\rho(B) \geq 1$: alors il existe $\lambda \in \mathbb{C}$ et $X \in \mathbb{C}^N$ tel que $BX = \lambda X$, $|\lambda| \geq 1$, $X \neq 0$ et donc

$$B^k X = \lambda^k X.$$

Mais $\left|\lambda^k\right| \nrightarrow 0$ quand $k \to \infty$. Donc B^k ne tend pas vers 0. La condition $\rho(B) < 1$ est donc nécessaire.
Supposons que $\rho(B) < 1$: Si $\|.\|$ est une norme sur $\mathbb{R}^N$ (ou $\mathbb{C}^N$), d'après (7.7), on a :

$$\left\|X^k - X\right\| \leq \|B\|^k \left\|X^0 - X\right\| \tag{7.9}$$

où $\|B\|$ est la norme matricielle induite de B. S'il existe une norme telle que $\|B\| < 1$, on a immédiatement

$$\lim_{k\to\infty} \left\|X^k - X\right\| = 0 \text{ pour tout choix de } X^0.$$

L'existence de cette norme résulte du lemme suivant :

Lemme 7.1 *Soit B une matrice carrée :*
(i) $\rho(B) \leq \|B\|$ pour toute norme matricielle $\|.\|$.
(ii) $\forall \varepsilon > 0, \exists \|.\|$ norme sur $\mathbb{R}^N$ telle que pour la norme matricielle induite $\|B\| \leq \rho(B) + \varepsilon$.

Démonstration : Le point (i) s'obtient comme suit : soit (λ, X) une valeur propre et un vecteur propre de B.

$$BX = \lambda X \Longrightarrow |\lambda| \|X\| \leq \|B\| \|X\| \Longrightarrow |\lambda| \leq \|B\|.$$

Puisque c'est vrai pour tout λ valeur propre, on en déduit $\rho(B) \leq \|B\|$.
La démonstration de (ii) est un peu plus longue. On peut la trouver dans de nombreux livres (par exemple "Introduction à l'analyse numérique matricielle et à l'optimisation", P.G. Ciarlet, Masson).

7.2.3.1 Vitesse de convergence

Une première estimation est obtenue à l'aide du lemme suivant :

Lemme 7.2 *Pour toute matrice carrée B, et toute norme matricielle*

$$\lim_{k\to\infty} \left\|B^k\right\|^{1/k} = \rho(B). \tag{7.10}$$

Démonstration :

$$\rho(B) = \left(\rho\left(B^k\right)\right)^{1/k} \leq \left\|B^k\right\|^{1/k} \tag{7.11}$$

ce qui donne une inégalité dans un sens.
Soit maintenant $\varepsilon > 0$ et $B_\varepsilon = \frac{B}{\rho(B)+\varepsilon}$; alors $\rho(B_\varepsilon) = \frac{\rho(B)}{\rho(B)+\varepsilon} < 1$. On en déduit

$$\lim_{k\to\infty} B_\varepsilon^k = 0 \text{ soit } \lim_{k\to\infty} \frac{B^k}{(\rho(B)+\varepsilon)^k} = 0.$$

Ainsi pour k assez grand et pour toute norme matricielle :

$$\left\|B^k\right\| \leq (\rho(B)+\varepsilon)^k \text{ ou } \left\|B^k\right\|^{1/k} \leq \rho(B)+\varepsilon. \tag{7.12}$$

De (7.12) et (7.11), on déduit le lemme.

Conséquence du lemme 7.2 : Nous avons vu (cf. (7.7)) que

$$\left\|X^k - X\right\| \leq \left\|B^k\right\| \left\|X^0 - X\right\|.$$

La vitesse de convergence de X^k vers X dépend donc de la vitesse de convergence vers 0 de B^k et donc de $\rho(B)$: il faut que $\rho(B)$ soit strictement inférieur à 1 et la vitesse de convergence sera d'autant plus rapide que $\rho(B)$ est petit. Ainsi, pour k assez grand, on a :

$$\left\|X^k - X\right\| \leq (\rho(B)+\varepsilon)^k \left\|X^0 - X\right\|.$$

De plus, si $\rho\left(\widetilde{B}\right) < \rho(B)$, la convergence de $\widetilde{X}^{k+1} = \widetilde{B}\widetilde{X}^k + \widetilde{d}$ sera plus rapide que celle de X^k. Pour comparer les vitesses de convergence on utilise parfois :

Définition 7.5 *On appelle taux asymptotique de convergence :* $R_\infty(B) = -\log(\rho(B))$.

Ainsi, si $\rho\left(\widetilde{B}\right) = \rho(B)^2 < 1$, on dit (et on vérifie) que la convergence de $\widetilde{X}^k$ est deux fois plus rapide que celle de X^k.

7.2.3.2 Application aux méthodes de Jacobi, Gauss-Seidel et Relaxation

Rappelons les "matrices B" de chaque méthode avec les notations de (7.5) :
Jacobi : $J := D^{-1}(E+F)$
Gauss-Seidel : $L_1 := (D-E)^{-1}F$
Relaxation : $L_\omega := (D-\omega E)^{-1}((1-\omega)D + \omega F)$.

Proposition 7.1 $\rho(L_\omega) \geq |\omega - 1|$. *Donc la méthode de relaxation ne peut converger que si*
$0 < \omega < 2$.

Démonstration : $|\det L_\omega| = |$produit des valeurs propres de $L_\omega| \leq \rho(L_\omega)^N$. Or

$$\det L_\omega = \frac{\det((1-\omega)D + \omega F)}{\det(D-\omega E)} = \frac{(1-\omega)^N \det D}{\det D} = (1-\omega)^N.$$

Ainsi

$$|1-\omega|^N \leq \rho(L_\omega)^N.$$

Nous allons maintenant examiner plusieurs cas de convergence de ces méthodes. Notons d'abord que, dès la dimension 2, elles peuvent être divergentes.

Exemple :

$$\begin{aligned} A &= \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix} \text{ (A est symétrique)} \\ D &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix} \\ J &= \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \quad \rho(J) = 2 \\ L_1 &= \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 0 & -4 \end{bmatrix} \quad \rho(L_1) = 4. \end{aligned}$$

On peut démontrer par exemple :

Théorème 7.3 *On suppose A symétrique définie positive. Alors la méthode de relaxation converge pour tout $\omega \in]0,2[$ (en particulier la méthode de Gauss-Seidel converge).*

On a aussi :

Théorème 7.4 *Si A est à diagonale strictement dominante i.e.*

$$\forall i = 1, ..., N \qquad |a_{ii}| > \sum_{j \neq i} |a_{ij}| \tag{7.13}$$

alors les méthodes de Gauss-Seidel et Jacobi sont convergentes.

On va utiliser pour le démontrer le

Lemme 7.3 (Hadamard-Gerschgörin) *Les valeurs propres d'une matrice quelconque A sont situées dans la réunion des disques*

$$D_i = \left\{ \lambda \in C; |\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \right\}, \; i = 1, ..., N.$$

Démonstration : Ecrivons $AX = \lambda X$ soit

$$\forall i = 1, ..., N, \; \sum_{j=1}^{N} a_{ij} x_j = \lambda x_i.$$

Soit k un indice tel que $|x_k| = \max_i |x_i|$. L'égalité ci-dessus avec $i = k$ implique

$$|(\lambda - a_{kk}) x_k| = \left| \sum_{j \neq i} a_{kj} x_j \right| \leq \sum_{j \neq k} |a_{kj}| \, |x_j|$$

soit

$$|\lambda - a_{kk}| \, |x_k| \leq \left(\sum_{j \neq k} |a_{kj}| \right) |x_k|$$

ce qui démontre le résultat.

Remarque 7.8 Le Lemme 7.3 a de nombreuses applications. Notons par exemple qu'il donne une estimation a priori sur la position des valeurs propres d'une matrice quelconque dans le plan complexe. Ce renseignement préliminaire peut être utile pour le calcul effectif de ces valeurs propres.

On en déduit aussi le

Corollaire 7.1 *Une matrice à diagonale strictement dominante est inversible.*

En effet, si A n'est pas inversible, 0 est valeur propre et d'après le lemme 7.3, il existe i tel que

$$|a_{ii}| \leq \sum_{j \neq i} |a_{ij}|$$

ce qui contredit la propriété de stricte dominance de la diagonale.

Démonstration du Théorème 7.4 : Calculons $\rho(J) = \rho\left(D^{-1}(E+F)\right)$. La matrice J est à diagonale nulle et pour $i \neq j$ on a

$$J_{ij} = -\frac{a_{ij}}{a_{ii}}.$$

D'après le lemme de Hadamard-Gerschgörin, si λ est valeur propre, on a

$$|\lambda - 0| = |\lambda| \leq \sum_{i \neq j} \frac{|a_{ij}|}{|a_{ii}|} < 1$$

et donc $\rho(J) < 1$. Soit maintenant λ une valeur propre de L_1 (éventuellement complexe). Si X est un vecteur propre associé, on a

$$\begin{aligned} (D-E)^{-1} FX &= \lambda X \Longrightarrow FX = \lambda (D-E) X \\ &\Longrightarrow \quad \forall i = 1, ..., N \qquad -\sum_{j>i} a_{ij} x_j = \lambda \sum_{j \leq i} a_{ij} x_j. \end{aligned}$$

Soit k un indice tel que $|x_k| = \max_i |x_i|$. La relation ci-dessus écrite avec $i = k$ donne

$$\begin{aligned} \lambda a_{kk} x_k &= -\sum_{j>k} a_{kj} x_j - \lambda \sum_{j<k} a_{kj} x_j \\ |\lambda| \, |a_{kk}| &\leq \sum_{j>k} |a_{kj}| + |\lambda| \sum_{j<k} |a_{kj}| . \end{aligned}$$

Si $|\lambda| \neq 0$, puisque A est à diagonale strictement dominante, on obtient

$$\begin{aligned} |\lambda| \sum_{j \neq k} |a_{kj}| &< \quad \textstyle\sum_{j>k} |a_{kj}| + |\lambda| \sum_{j<k} |a_{kj}| \\ &\Longrightarrow \quad |\lambda| < 1 \text{ et donc } \rho(L_1) < 1. \end{aligned}$$

Remarque 7.9 On peut montrer que si $D^{-1}E$ et $D^{-1}F$ sont à éléments positifs ou nuls, alors, dès que la méthode de Jacobi est convergente, il en est de même de celle de Gauss-Seidel. De plus, la convergence de celle-ci est au moins aussi rapide. Ce phénomène n'est pas néanmoins systématique. Cependant, pour de nombreux systèmes, surtout lorsqu'ils

proviennent de la discrétisation d'équations aux dérivées partielles, la méthode de Gauss-Seidel converge plus vite que celle de Jacobi. De plus, pour une large famille de tels systèmes, on peut montrer l'existence d'un paramètre ω^* optimal pour lequel la méthode de relaxation est considérablement plus efficace : le nombre d'itérations nécessaires pour atteindre une précision donnée chute considérablement lorsque ω est voisin de ω^*.

Nous allons énoncer sans démonstration un résultat dans ce sens. Pour cela, nous avons besoin de la définition suivante : A étant décomposée comme précédemment, on note :

$$L = D^{-1}E, \quad U = D^{-1}F$$

d'où

$$\begin{aligned} A &= D\left(I - L - U\right), \quad J = L + U \\ L_\omega &= \left(I - \omega L\right)^{-1}\left(\left(1-\omega\right)I + \omega U\right). \end{aligned}$$

Définition 7.6 *On dira que A est de type (V), si pour $\alpha \neq 0$, les valeurs propres de $J(\alpha) = \alpha L + \frac{1}{\alpha}U$ sont indépendantes de α.*

Remarque 7.10 On montre que les matrices tridiagonales sont de type (V). De nombreuses matrices provenant de la discrétisation d'équations aux dérivées partielles sont également de type (V). Cette notion a été introduite par Varga qui utilise la terminologie "consistently ordered matrices" dont la traduction littérale française est peu heureuse.

On a alors le résultat général suivant :

Théorème 7.5 *Soit A de type (V). Alors*
(i) $\rho(L_1) = \rho(J)^2$.
Donc la méthode de Gauss-Seidel converge si et seulement si celle de Jacobi converge et elle converge alors deux fois plus vite.
(ii) Si, de plus, les valeurs propres de J sont réelles et $\rho(J) < 1$, alors le graphe de $\rho(L_\omega)$ a l'allure suivante :
Plus précisément, on a :

$$\omega^* = \frac{2}{1+\sqrt{1-\rho(J)^2}} \qquad \rho(L_{\omega^*}) = \left(\frac{\rho(J)}{1+\sqrt{1-\rho(J)^2}}\right)^2$$

et

$$\rho(L_\omega) = \begin{cases} \omega - 1 & si\ \omega^* \leq \omega \leq 2 \\ 1 - \omega + \frac{1}{2}\omega^2\rho(J)^2 + \omega\rho(J)\sqrt{1-\omega+\frac{1}{4}\omega^2\rho(J)^2} & si\ 0 \leq \omega \leq \omega^*. \end{cases}$$

Remarque 7.11 Le plus souvent, ω^* est déterminé expérimentalement. Noter à ce propos que la dérivée à gauche de $\rho(L_\omega)$ en $\omega = \omega^*$ est infinie. On aura donc plutôt intérêt à surévaluer ω^* car une erreur à droite sur ω^* sera moins sensible qu'une erreur à gauche.

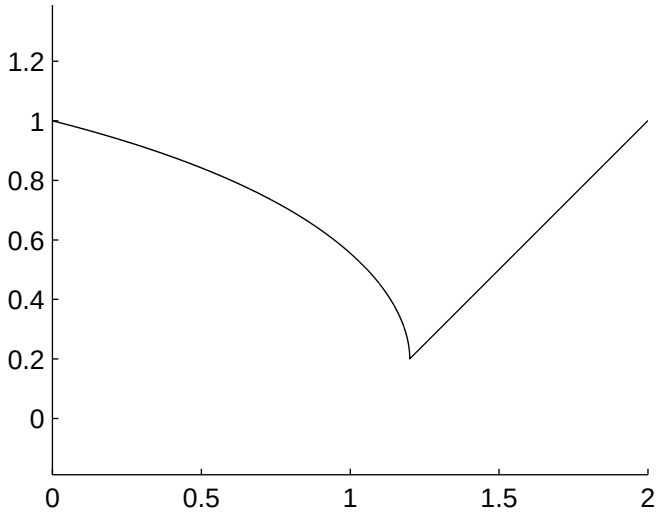

FIGURE 7.2 – Graphe du rayon spectral de L_ω

7.2.4 Application à la méthode des différences finies

Nous allons expliciter les résultats du théorème 7.5 pour deux systèmes linéaires provenant de la discrétisation par différences finies d'équations aux dérivées partielles simples.
Considérons d'abord le problème aux limites monodimensionnel standard

$$\begin{cases} -u''(x) = f(x) \quad 0 < x < 1 \\ u(0) = u(1) = 0. \end{cases} \tag{7.14}$$

On montre facilement que si f est continue sur [0,1], ce problème admet une solution u unique. Le calcul numérique de cette solution par la méthode des différences finies consiste à remplacer la recherche de u par celle d'un vecteur $u_h = (u_1, u_2, ..., u_{\mathrm{N}})$ représentant les valeurs d'une solution approchée en les points $x_1, x_2, ..., x_{\mathrm{N}}$ d'une subdivision de l'intervalle [0,1] que nous supposerons régulière pour simplifier soit

$$x_i = ih,\ i = 0, ..., N+1,\ \ h = \frac{1}{N+1}.$$

Les valeurs en $x_0 = 0$ et $x_{\mathrm{N}+1} = 1$ seront supposées nulles pour satisfaire aux conditions aux limites de (7.14). Le problème (7.14) est alors remplacé par un problème approché où $u''(x_i)$ est remplacé par la différence finie

$$u''(x_i) \approx \frac{u(x_{i+1}) + u(x_{i-1}) - 2u(x_i)}{h^2}.$$

Le vecteur approché u_h cherché est donc défini par le système

$$\begin{cases} -\dfrac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = f_i \quad i = 1, ..., N \\ u_0 = u_{\mathrm{N}+1} = 0 \end{cases} \tag{7.15}$$

où on a noté $f_i = f(x_i)$. Tenant compte des conditions aux limites $u_0 = 0$ dans la première équation et $u_{N+1} = 0$ dans la dernière, il s'agit donc de résoudre le système linéaire :

$$\frac{1}{h^2} A u_h = f_h \tag{7.16}$$

où $f_h = (f_1, ..., f_N)$ et

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix}. \tag{7.17}$$

On vérifie que A est une matrice tridiagonale symétrique définie positive. Le système (7.15) pourra être résolu à l'aide de la méthode directe développée dans le théorème 6.1 conduisant à un nombre d'opérations de l'ordre de $8N$. Cette méthode est d'ailleurs fortement conseillée.

Cependant, à titre indicatif, nous allons expliciter les résultats du théorème 7.5 sur cet exemple. La matrice A étant définie par (7.17), on a, avec les notations utilisées précédemment

$$J(\alpha) = \alpha D^{-1} E + \frac{1}{\alpha} D^{-1} F = \frac{1}{2} \begin{bmatrix} 0 & \frac{1}{\alpha} & 0 & \cdots & \cdots & 0 \\ \alpha & \ddots & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \frac{1}{\alpha} \\ 0 & \cdots & \cdots & 0 & \alpha & 0 \end{bmatrix}.$$

On vérifie aisément que le vecteur $U^k = \left(U_i^k\right)_{1 \le i \le N}$ tel que

$$U_i^k = \alpha^{i-1} \sin i \frac{k\pi}{N+1}, \ i = 1, ..., N$$

est vecteur propre de $J(\alpha)$ pour la valeur propre $\lambda_k = \cos \frac{k\pi}{N+1}$ et ce pour $k = 1, ..., N$. Ceci montre que A est de type (V) et

$$\rho(J) = \max_{1 \le k \le N} |\lambda_k| = \cos \frac{\pi}{N+1}.$$

On en déduit en particulier que, lorsque N est grand,

$$\rho(J) = 1 - \frac{1}{2} \frac{\pi^2}{(N+1)^2} + O\left(\frac{1}{N^4}\right) \tag{7.18}$$

et donc $\rho(J)$ tend vers 1 lorsque N tend vers l'infini. La vitesse de convergence décroît ainsi avec N. D'après le théorème 7.5, on a

$$\rho(L_1) = \rho(J)^2 = 1 - \frac{\pi^2}{(N+1)^2} + O\left(\frac{1}{N^4}\right) \tag{7.19}$$

$$\omega^* = \frac{2}{1+\sqrt{1-\cos^2\left(\frac{\pi}{N+1}\right)}} = \frac{2}{1+\sin\left(\frac{\pi}{N+1}\right)} \tag{7.20}$$

$$\rho(L_{\omega^*}) = \frac{\cos^2\left(\frac{\pi}{N+1}\right)}{\left(1+\sin\left(\frac{\pi}{N+1}\right)\right)^2}. \tag{7.21}$$

Ainsi, pour N grand

$$\rho(L_{\omega^*}) = 1 - \frac{2\pi}{N+1} + O\left(\frac{1}{N^2}\right) \tag{7.22}$$

Les relations (7.18), (7.19), (7.22) permettent de comparer les vitesses de convergence des différentes méthodes lorsque N est grand. Rappelons que, comme conséquence du lemme 7.2, si x_{N} est définie par

$$X_{n+1} = Bx_{\mathrm{N}} + d$$

alors, asymptotiquement

$$\|x_{\mathrm{N}} - X_\infty\| \leq \rho(B)^n \|X_0 - X_\infty\|. \tag{7.23}$$

Notons $c_0 = \|X_0 - X_\infty\|$; pour rendre $\|x_{\mathrm{N}} - X_\infty\|$ inférieur à ε, il faut, en supposant l'estimation (7.23) optimale ce qui est le plus souvent le cas, que $c_0\rho(B)^n \leq \varepsilon$, soit en notant $\alpha = -\log\frac{\varepsilon}{c_0}$ et $R_B = -\log\rho(B)$ (vitesse de convergence)

$$n \geq \frac{\alpha}{R_B}.$$

Pour de grandes dimensions N, d'après (7.18), (7.19), (7.22), on a

$$R_J \sim \frac{\pi^2}{2N^2},\ R_{L_1} \sim \frac{\pi^2}{N^2},\ R_{L_{\omega^*}} \sim \frac{2\pi}{N}.$$

Il sera donc nécessaire de prendre

$$\begin{aligned} n &\sim \frac{2\alpha}{\pi^2}N^2 \text{ pour Jacobi} \\ n &\sim \frac{\alpha}{\pi^2}N^2 \text{ pour Gauss-Seidel} \\ n &\sim \frac{\alpha}{2\pi}N \text{ pour la relaxation optimale.} \end{aligned}$$

Cette dernière méthode est donc plus que N fois plus rapide que celles de Jacobi et Gauss-Seidel.

Afin d'évaluer le temps global de calcul pour éventuellement comparer avec la méthode directe suggérée plus haut, outre le nombre d'itérations, il est nécessaire de tenir compte

du nombre d'opérations élémentaires à chaque itération, soit kN où k est une constante de l'ordre de 8. D'autre part, il faut aussi tenir compte d'un choix "raisonnable" de la tolérance ε. Une analyse de l'erreur de discrétisation montre que

$$\max_i |u(x_i) - u_i| \leq Ch^2$$

où u est la solution exacte, (u_i) la solution approchée et C une constante dépendant de u. Il est donc raisonnable de choisir ε d'ordre h^2, soit, pour une certaine constante a

$$\alpha \approx -\log ah^2 \approx \log aN^2 \sim 2\log N \text{ pour } N \text{ grand.}$$

On obtient alors le tableau suivant valable pour N grand

Méthode	Nombre d'itérations	Nombre d'opérations
Jacobi	$\frac{4}{\pi^2}N^2 \log N$	$k\frac{4}{\pi^2}N^3 \log N$
Gauss-Seidel	$\frac{2}{\pi^2}N^2 \log N$	$k\frac{2}{\pi^2}N^3 \log N$
Relaxation optimale	$\frac{1}{\pi}N \log N$	$k\frac{1}{\pi}N^2 \log N$

Comme annoncé, on constate que le nombre d'opérations requis est bien supérieur à celui requis pour la méthode directe du théorème 6.1, soit un facteur de N. Cette différence importante est bien sûr très liée à la structure tridiagonale. Elle devient moins importante pour de grandes matrices à bandes très creuses. De plus, dans ce cas, même si le nombre d'opérations est plus grand, pour des raisons de stockage en mémoire, on pourra préférer une méthode itérative à une factorisation qui "remplit" les bandes et rend ainsi le stockage des matrices-facteurs difficile, voire impossible pour des matrices très grandes.

Pour terminer ce paragraphe, nous énonçons sans détails les résultats relatifs au système discrétisé associé à l'équation aux dérivées partielles bidimensionnelle :

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2}(x,y) - \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \text{ sur } \Omega \subset \mathbb{R}^2 \\ u(x,y) = 0 \text{ sur le bord de } \Omega. \end{cases}$$

Nous supposons que Ω est le carré $[0,1]\times[0,1]$. On le munit d'un maillage uniforme de même pas h dans les deux directions dont les noeuds sont les points (ih, jh), $i,j = 0,1,...,N+1$. Le problème discrétisé consiste à trouver une approximation de la solution aux noeuds du maillage, la dérivée seconde en un point (ih,jh) étant remplacée par la formule de différences finies à 5 points

$$u''(ih,jh) \approx \frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2}$$

où $U_{i,j}$ est la valeur de la solution au point (ih,jh). Le système discrétisé s'écrit alors avec $f_{i,j} = f(ih,jh)$ et $h = \frac{1}{N+1}$:

$$\begin{cases} -\frac{U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{i,j}}{h^2} = f_{i,j} \quad i,j = 1,...,N \\ U_{0,j} = U_{N+1,j} := 0, \ \ j = 0,...,N+1 \\ U_{i,0} = U_{i,N+1} := 0, \quad i = 0,...,N+1 \end{cases}$$

L'écriture matricielle de ce système nécessite le choix d'une numérotation des noeuds. Une numérotation par ligne conduit à un système

$$\frac{1}{h^2}AU_h = f_h$$

où

$$A = \begin{bmatrix}
4 & -1 & 0 & \cdots & 0 & -1 & & & & & & & \\
-1 & 4 & -1 & 0 & \cdots & 0 & \ddots & & & & & & \\
0 & -1 & 4 & -1 & 0 & \cdots & 0 & -1 & & & & & \\
-1 & & & 4 & -1 & & & & \ddots & & & & \\
 & \ddots & & -1 & \ddots & -1 & & & & \ddots & & & \\
 & & -1 & & -1 & 4 & & & & & \ddots & & \\
 & & & \ddots & & & \ddots & & & \ddots & & -1 & \\
 & & & & \ddots & & \ddots & \ddots & & & \ddots & & \ddots \\
 & & & & & \ddots & & & & \ddots & \ddots & & -1 \\
 & & & & & & -1 & & & 4 & & -1 & \\
 & & & & & & & \ddots & & -1 & & \ddots & -1 \\
 & & & & & & & & -1 & 0 & \cdots & 0 & -1 & 4
\end{bmatrix}$$

on montre que A est de type (V). On vérifie que les vecteurs

$$U^{k,l},\ k,l = 1,...,N$$

de composantes

$$U^{k;l}_{i,j} = \sin i\frac{k\pi}{N+1}\sin j\frac{l\pi}{N+1},\ i,j = 1,...,N$$

sont vecteurs propres de $J = L + U$ pour les valeurs propres

$$\lambda^{k,l} = \frac{1}{2}\left(\cos\frac{k\pi}{N+1} + \cos\frac{l\pi}{N+1}\right).$$

Ainsi

$$\rho(J) = \max_{k,l}\left|\lambda^{k,l}\right| = \cos\frac{\pi}{N+1}.$$

On retrouve donc les mêmes formules qu'en (7.18), (7.19), (7.22). Pour atteindre une précision en h^2, le même nombre d'itérations que dans l'exemple précédent est nécessaire. Chaque itération requiert environ kN^2 opérations, d'où un coût total d'environ (à un facteur près) $N^4 \log N$ pour Jacobi et Gauss-Seidel, $N^3 \log N$ pour la relaxation optimale. A titre de comparaison, une factorisation de Cholesky nécessiterait environ $\frac{1}{2}N^4$ opérations.

7.3 Autres méthodes itératives modernes

Les méthodes itératives de type "point fixe" que nous venons d'étudier sont les premières méthodes introduites dans ce cadre. Elles sont néanmoins souvent limitées en robustesse

pour les grands problèmes. Elles permettent toutefois d'être utiles dans le cadre du préconditionnement. Des méthodes itératives plus robustes existent. Citons à titre d'exemple la méthode du gradient conjugué (cf. chapitre 9) ou encore les méthodes par sous-espace de Krylov (GMRES, BiCGStab,...). Ces méthodes, bien que très efficaces, sont inutilisables en pratique si un préconditionneur adéquat n'est pas associé (factorisation LU incomplète, multigrille, SPAI...).

7.4 Exercices du chapitre 7

Exercice 7.1 Montrer que $\max_{i,j} |a_{ij}|$ n'est pas une norme matricielle, mais que $\left(\sum_{i,j} a_{ij}^2\right)^{1/2}$ en est une.

Exercice 7.2 Montrer $\rho(A^*A) = \rho(AA^*)$.

Exercice 7.3 Montrer que $\rho(J) < 1 < \rho(L_1)$ lorsque

$$A = \begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}.$$

Exercice 7.4 On considère la matrice

$$A = \begin{pmatrix} 2 & 2\sqrt{2} \\ 2\sqrt{2} & 9 \end{pmatrix}.$$

Quel est son conditionnement pour les normes matricielles usuelles ? On multiplie la première ligne par un coefficient α, comment choisir α pour que le conditionnement soit minimum ?

7.5 TD7 : méthodes itératives

Exercice 1. On souhaite résoudre un système linéaire

$$A\mathbf{x} = \mathbf{b}$$

de manière itérative en considérant une méthode de Jacobi ou de Gauss-Seidel. Nous notons J et L_1 les matrices respectives d'itérations. Soit A la matrice donnée par

$$A = \begin{pmatrix} 1 & -1 \\ 2 & 4 \end{pmatrix} \tag{7.24}$$

1) Donner les matrices J et L_1 (en détaillant leur obtention) correspondant à (7.24).
2) Ces deux méthodes itératives convergent-elles ? (Conseil : étudier le rayon spectral de J et L_1.)

Exercice 2. 1. Programmer simplement la méthode de Gauss-Seidel. Reprenez la matrice et le second membre de l'exercice 2 de la feuille de TD 6 de la semaine dernière. Etudier alors la convergence de la méthode en regardant notamment l'évolution du résidu $\|Au - b\|/\|b\|$.
Fonctions Matlab utile : tril, triu

2. Matlab propose six méthodes itératives pour la résolution de problèmes linéaires : *bicg, bicgstab, cgs, gmres, pcg, qmr.*
Ces six fonctions s'utilisent sous Matlab d'une façon semblable, considérons par exemple la fonction *cgs*. Il est possible de la lancer comme ceci :

$$[u, flag, relres, iter, resvec] = cgs(A, b, tol, maxit)$$

A est la matrice du système,
b est le second menbre,
tol est la tolérance sur la quantité $\|Au - b\|/\|b\|$ (test d'arrêt de la méthode),
$maxit$ est le nombre maximum d'itérations,
u est la solution approchée fournie par Matlab,
$flag$ "drapeau" permet de savoir si la méthode a convergé, et la nature de l'échec le cas échéant,
$relres$ dernier résidu $\|Au - b\|/\|b\|$,
$iter$ nombre d'itérations nécessaires,
$resvec$ est un vecteur contenant la valeur du rapport $\|Au - b\|/\|b\|$ au cours des différentes itérations.

Résoudre le système $Au = b$ par cette méthode *cgs* (réaliser plusieurs essais pour différentes valeurs de n et de tol). Qu'observe-t-on ?
Fonction Matlab utile : semilogy.

7.6 Aide-mémoire : Rappels sur les valeurs propres

Soit A une matrice carrée de dimension n. On appelle valeur propre et vecteur propre de A un couple (λ, U) avec $\lambda \in \mathbb{R}$ (ou $\lambda \in \mathbb{C}$) et $U \in \mathbb{R}^n$ (ou $U \in \mathbb{C}^n$) tels que

$$U \neq 0 \qquad \text{et} \quad AU = \lambda U\,.$$

Comme le vecteur propre U est clairement dans le noyau de la matrice $A - \lambda Id$ (le noyau d'une matrice ou d'une application linéaire est l'ensemble des vecteurs dont l'image est 0), les valeurs propres λ peuvent être caractérisées comme les nombres tels que le déterminant de $A - \lambda Id$ est nul. Or l'équation

$$\det\,(A - \lambda Id) = 0$$

est une équation polynômiale de degré n en λ ; donc la recherche des valeurs propres revient théoriquement à rechercher les zéros d'un polynôme. Dans la pratique, ce n'est pas ainsi qu'on procède, car ce serait beaucoup trop lourd.

On appelle **spectre** de A, l'ensemble de ses valeurs propres. Le rayon spectral de A est le module de la plus grande d'entre elles, on le note $\rho(A)$:

$$\rho(A) = \max\{|\lambda|,\ \lambda \text{ valeur propre de } A\}.$$

Remarquons que les valeurs propres de A^2 sont exactement les carrés des valeurs propres de A, les valeurs propres de A^n sont exactement les puissances n-ièmes des valeurs propres de A, les valeurs propres de A^{-1} sont exactement les inverses des valeurs propres de A.

On voit donc qu'une matrice A de dimension n a exactement n valeurs propres, éventuellement complexes, éventuellement multiples. On dit que A est **diagonalisable** s'il existe une base de vecteurs propres. Si toutes les valeurs propres sont simples, la matrice est diagonalisable (au moins sur $\mathbb{C}$). Si certaines valeurs propres sont multiples, il est possible qu'elle ne le soit pas.

Si la matrice A est symétrique (ou plus généralement si A est normale, c'est-à-dire commute avec sa transposée ou son adjointe), alors toutes ses valeurs propres sont réelles et A est diagonalisable.

La somme de toutes les valeurs propres de la matrice A est égale à la trace de A, c'est-à-dire la somme de ses termes diagonaux $\sum_{i=1}^{n} a_{i,i}$. Le produit de toutes les valeurs propres de la matrice A est égal au déterminant de A.

Chapitre 8

Résolution d'équations et systèmes non linéaires

8.1 Introduction

Le but de ce chapitre est de décrire les algorithmes les plus fréquemment utilisés pour résoudre des équations non linéaires du type :

$$f(x) = 0$$

où

1. x est une variable réelle, f une fonction à valeurs réelles
2. x est un vecteur de $\mathbb{R}^n$, f une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^n$.

Le premier point sera le plus détaillé : la convergence des algorithmes est analysée. On présentera également des techniques d'accélération de convergence. Pour les fonctions de $\mathbb{R}^n$ dans $\mathbb{R}^n$ on présentera rapidement les algorithmes qui seront en fait inspirés de la dimension 1.

8.2 Cas des fonctions d'une variable

8.2.1 Préliminaires : séparation des zéros

Exemple 8.1 résoudre $x - 0,2\sin x - 0,5 = 0$, x réel.

Exemple 8.2 résoudre $\cos x = e^{-x}$, x réel.

Le premier travail consiste en une analyse mathématique minimale pour séparer les zéros, c'est-à-dire déterminer des intervalles $[a_i, b_i]$ dans lesquels l'équation considérée a une solution et une seule.
La méthode la plus simple est d'utiliser qu'une fonction continue strictement monotone sur un intervalle $[a_i, b_i]$ et telle que $f(a_i) f(b_i) \leq 0$ a un zéro et un seul dans l'intervalle $[a_i, b_i]$. Par exemple :

$$f_1(x) = x - 0,2\sin x - 0,5$$

$$f_1'(x) = 1 - 0,2\cos x \geq 0 \text{ pour tout } x\ .$$

Donc f_1 est strictement croissante sur $\mathbb{R}$. Comme $f_1(0) = -0,5 < 0$, $f_1(\pi) = \pi - 0,5 > 0$, f_1 a un unique zéro dans $[0, \pi]$.
Un calcul similaire pour l'exemple 8.2 conduit à une impasse si on pose $f_2(x) := \cos x - e^{-x}$ puisque la dérivée $f_2'(x) = \sin x + e^{-x}$ est du même type. Cette situation est bien sûr beaucoup plus fréquente. On peut alors :

- choisir plus judicieusement la fonction auxiliaire f_2. Ici, on peut par exemple poser $f_2(x) := e^x \cos x - 1$. Les zéros de f_2 sont exactement les solutions cherchées. D'autre part :
$$f_2'(x) = e^x(\cos x - \sin x) = \sqrt{2}e^x \cos\left(x - \frac{\pi}{4}\right).$$
Ainsi f_2 est strictement monotons sur les intervalles du type $\left[\frac{\pi}{4} + k\frac{\pi}{2}, \frac{\pi}{4} + (k+1)\frac{\pi}{2}\right]$, k entier. L'étude des signes successifs de $f\left(\frac{\pi}{4} + k\frac{\pi}{2}\right)$ permet alors de localiser les zéros éventuels.
- On peut procéder par tatonnements en s'aidant au maximum d'informations complémentaires disponibles. Ainsi, dans l'exemple 8.2, le tracé des représentations graphiques des fonctions $x \longmapsto \cos x$ et $x \longmapsto e^{-x}$ est particulièrement suggestif.

Dans la suite, nous nous placerons le plus souvent dans le cas où $f : [a, b] \to \mathbb{R}$ admet un unique zéro dans l'intervalle $[a, b]$.

8.2.2 Quelques algorithmes classiques

8.2.2.1 Méthode de dichotomie (ou bisection)

Reprenons l'exemple 8.1. Nous avons vu que si $f(x) = x - 0.2\sin x - 0.5$, $f(0) < 0$, $f(\pi) > 0$; d'où un zéro dans $(0, \pi)$. Si on calcule la valeur de f en $\left(\frac{\pi}{2}\right)$, on constate que $f\left(\frac{\pi}{2}\right) = \frac{\pi}{2} - 0,7 > 0$. Donc ce zéro est en fait dans $\left[0, \frac{\pi}{2}\right]$. On peut alors calculer $\mathrm{f}\left(\frac{\pi}{4}\right) = 0,14 > 0$, qui montre qu'il est dans $\left[0, \frac{\pi}{4}\right]$.
Il est clair qu'une répétition de ce procédé donne un encadrement de plus en plus précis du zéro cherché et fournit donc un algorithme de calcul de ce zéro.

Algorithme 8.1 Soit $f : [a_0, b_0] \to R$ continue et telle que $f(a_0) f(b_0) \leq 0$.

$$\left[\begin{array}{c} \text{Pour } n = 0, 1, 2, ..., N, \text{ faire} \\ m := \frac{(a_n + b_n)}{2} \\ \text{Si } f(a_n) f(m) \leq 0,\ a_{n+1} := a_n,\ b_{n+1} := m \\ \text{Sinon} \qquad\qquad a_{n+1} := m,\ b_{n+1} := b_n. \end{array}\right.$$

On a : $a_{n+1} - b_{n+1} = \frac{1}{2}(a_n - b_n)$, soit par récurrence $a_n - b_n = \frac{1}{2^n}(a_0 - b_0)$. Il en résulte que cette méthode est toujours convergente puisque $a_n - b_n$ tend vers 0 quand n tend vers l'infini. On peut choisir le temps d'arrêt N pour que :

$$\frac{1}{2^N}(a_0 - b_0) < \varepsilon = \text{ précision choisie.}$$

On obtient alors un encadrement de la solution cherchée à la précision voulue. Bien que cette situation soit très alléchante, on verra qu'en fait cette méthode converge plutôt lentement comparée à celles qui suivent.

Remarque 8.1 Dans l'exemple 2.1, on a $f(0) = -0,5$, $f(\pi) = 2,64$. Ceci laisse immédiatement penser que le zéro cherché doit être plus près de 0 que de π. Le plus efficace n'est donc pas de tester la valeur de f en le milieu de $[0, \pi]$ mais plutôt en un point plus voisin de 0 tenant compte de cette différence de poids entre $f(0)$ et $f(\pi)$. Ceci conduit aux algorithmes dits de "fausse position" qui convergent généralement plus vite. Nous ne les expliciterons pas ici.

Remarque 8.2 La méthode ci-dessus converge même si f a plusieurs zéros dans l'intervalle $[a_0, b_0]$.

8.2.2.2 Méthode de la sécante

Soit f admettant un zéro dans l'intervalle $[x_{-1}, x_0]$. Pour obtenir une première approximation de ce zéro, l'idée est de remplacer f par son interpolé linéaire sur $[x_{-1}, x_0]$, soit par

$$Y(x) = f(x_0) + (x - x_0) \frac{f(x_0) - f(x_{-1})}{x_0 - x_{-1}},$$

l'unique fonction linéaire dont les valeurs coïncident avec celles de f en x_{-1} et x_0. L'approximation x_1 est alors obtenue en résolvant :

$$Y(x_1) = 0$$

soit

$$x_1 = x_0 - f(x_0) \frac{x_0 - x_{-1}}{f(x_0) - f(x_{-1})}.$$

Géométriquement (cf Figure 8.1), ceci revient à remplacer la courbe d'équation $y = f(x)$

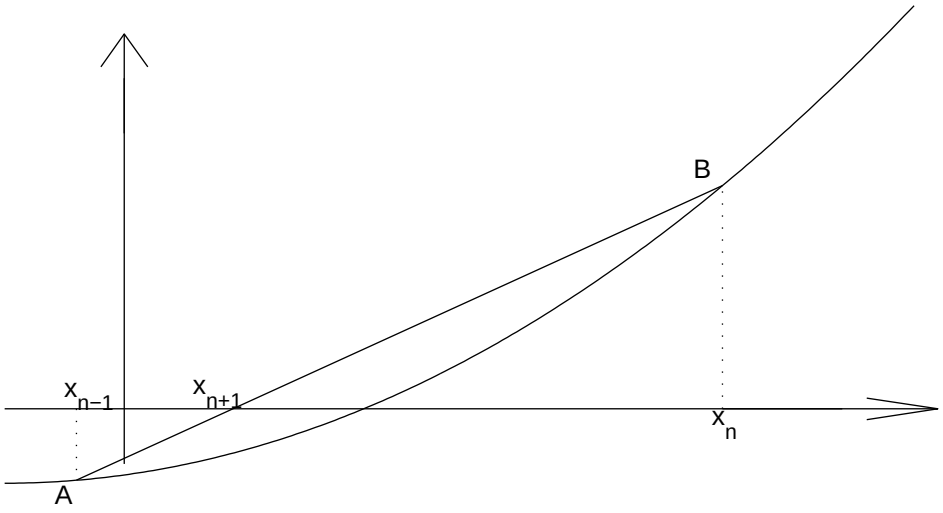

FIGURE 8.1 – Méthode de la sécante

par la sécante AB et x_{n+1} est l'intersection de AB avec la droite (Ox).

Comme le montre le dessin, x_{n+1} semble plus voisin du zéro cherché que x_{n-1} ou x_n. Pour trouver une meilleure approximation, il suffit de répéter le procédé à l'aide des points (x_n, x_{n+1}). En continuant, on obtient l'algorithme suivant.

Algorithme 8.2 x_0 et x_{-1} étant donnés,

$$\left[\begin{array}{c} \text{Pour } n = 0, 1, 2, ... \\ x_{n+1} = x_n - f(x_n) \dfrac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}. \end{array}\right.$$

8.2.2.3 Critère d'arrêt

Cet algorithme n'est évidemment pas complet tant qu'on n'a pas précisé un critère d'arrêt. Nous verrons plus loin que, généralement, x_n converge vers la solution x_∞ cherchée ce qui signifie que pour n grand, x_n est voisin de x_∞. Sans plus d'informations, il est difficile de savoir ce que signifie numériquement "n grand", et c'est là une difficulté fréquente en analyse numérique.
Un critère d'arrêt souvent utilisé consiste à choisir *a priori* une tolérance ε et à terminer l'algorithme lorsque

$$|x_{n+1} - x_n| \leq \varepsilon\,. \tag{8.1}$$

Ce n'est pas complètement satisfaisant ; pour s'en convaincre, on peut penser à la suite :

$$x_n = 1 + \frac{1}{2} + ... + \frac{1}{n}$$

qui vérifie $\lim_{n\to\infty} |x_{n+1} - x_n| = 0$ bien que tendant vers l'infini.
Puisqu'on veut résoudre $f(x) = 0$, un autre critère d'arrêt possible consiste à s'arrêter lorsque

$$|f(x_n)| < \varepsilon\,. \tag{8.2}$$

Un numéricien particulièrement scrupuleux pourra décider de ne s'arrêter que lorsque **les deux** critères d'arrêts (8.1) et (8.2) sont **simultanément** vérifiés. Dans le cas de l'algorithme de la sécante ci-dessus, on peut aussi utiliser :

$$|f(x_n) - f(x_{n-1})| < \varepsilon\,.$$

ce critère d'arrêt a l'avantage d'éviter une possible division par 0.
Enfin, pour éviter à l'ordinateur de tourner sans s'arrêter lorsqu'il n'y a pas convergence, il est évidemment **indispensable de toujours** mettre un critère limitant le nombre total d'itérations.

8.2.2.4 Méthode de Newton

Ici, au lieu d'assimiler la courbe "$y = f(x)$" à une sécante, on l'assimile à la tangente en un point $(x_n, f(x_0))$ soit, la droite d'équation :

$$Y = f(x_n) + f'(x_n)(x - x_n)\,.$$

Son intersection avec l'axe des abscisses fournit une approximation de la solution x_∞. A nouveau, on renouvelle le procédé jusqu'à obtenir une approximation suffisante, d'où l'algorithme :

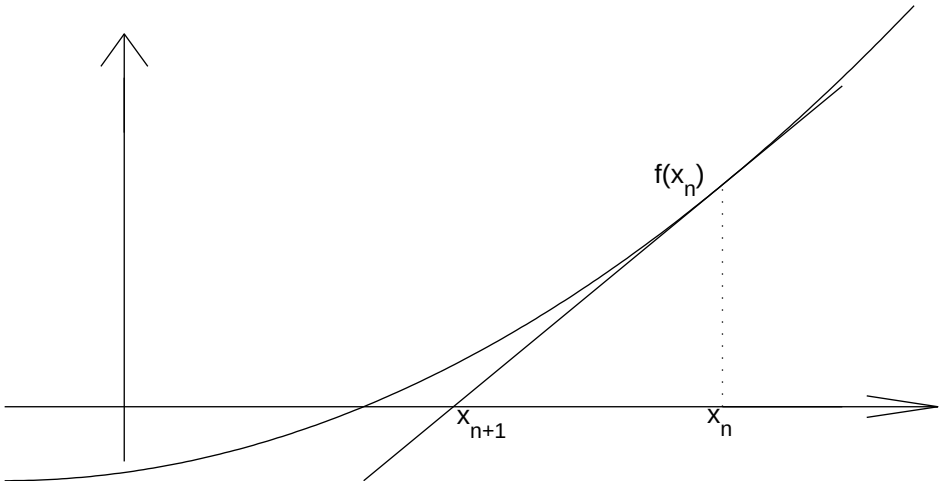

FIGURE 8.2 – Méthode de Newton

Algorithme 8.3 x_0 étant donné

$$\left[\begin{array}{l} \text{Pour } n = 0, 1, 2, ... \\ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \end{array}\right.$$

Remarque 8.3 Cet algorithme est initié à l'aide d'un seul point. Il peut être vu comme une modification de l'algorithme précédent où le quotient $\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$ a été remplacé par $\frac{1}{f'(x_n)}$. Nous verrons plus loin qu'il donne lieu à une convergence beaucoup plus rapide.

8.2.2.5 Méthode de point fixe

Elle consiste à d'abord remplacer l'équation

$$(*) \quad f(x) = 0$$

par une équation

$$(**) \quad g(x) = x$$

ayant les mêmes solutions. On est ainsi ramené à la recherche des points fixes de l'application g. Le remplacement de $(*)$ par $(**)$ est toujours possible en posant, par exemple, $g(x) = f(x) + x$. Ce n'est évidemment pas forcément le meilleur choix !

Exemple 8.3 Pour l'équation

$$x^2 - x - 2 = 0$$

on peut prendre

$$\begin{aligned} g(x) &= x^2 - 2 \\ g(x) &= \sqrt{2+x} \\ g(x) &= 1 + \frac{2}{x} \\ g(x) &= x - \frac{x^2 - x - 2}{m} \text{ pour tout paramètre } m \neq 0. \end{aligned}$$

l'équation étant sous la forme $(**)$, on a alors l'algorithme suivant :

Algorithme 8.4 On choisit x_0

$$\left[\begin{array}{l} \text{Pour } n = 0, 1, 2, ... \\ \quad x_{n+1} = g(x_n). \end{array} \right.$$

Cette méthode est justifiée par la :

Proposition 8.1 *Soit $g : [a,b] \to [a,b]$ continue et $x_0 \in [a,b]$. Si x_n converge vers x_∞, alors*

$$x_\infty = g(x_\infty).$$

La démonstration est immédiate en passant à la limite dans l'égalité $x_{n+1} = g(x_n)$ et en utilisant la continuité de g au point x_∞.

Une difficulté majeure dans l'algorithme 8.4 est qu'il peut arriver qu'on ne puisse calculer $g(x_n)$. Il suffit de penser à ce qui se passe lorsque $g(x) = -\sqrt{x}$!

Remarque 8.4 Il peut être intéressant de construire graphiquement les valeurs successives de x_n dans certains cas : prenons $g(x) = \sqrt{2+x}$ comme dans l'exemple 8.3 :

On utilise la première bissectrice pour reporter les valeurs successives de x_n. On peut se convaincre à l'aide de ce type de construction que certains points fixes ne sont jamais atteints par l'algorithme 8.4 (cf Figure 8.4 ci après) :

Exemple 8.4 $g(x) = x^2$ Si x_0 est choisi à gauche de 1, la suite x_n converge vers le point fixe 0. Si x_0 est choisi à droite de 1, la suite x_n tend vers l'infini. A moins de choisir exactement $x_0 = 1$, on voit que la suite ne converge jamais vers 1 : c'est un point fixe instable. On y reviendra dans le paragraphe suivant.

Cet exemple met par ailleurs en évidence l'influence du choix de x_0 sur le comportement de x_n.

8.2.3 Convergence des algorithmes

8.2.3.1 Méthodes de point fixe

Commençons par traiter le cas du point fixe qui est fondamental d'un point de vue théorique.

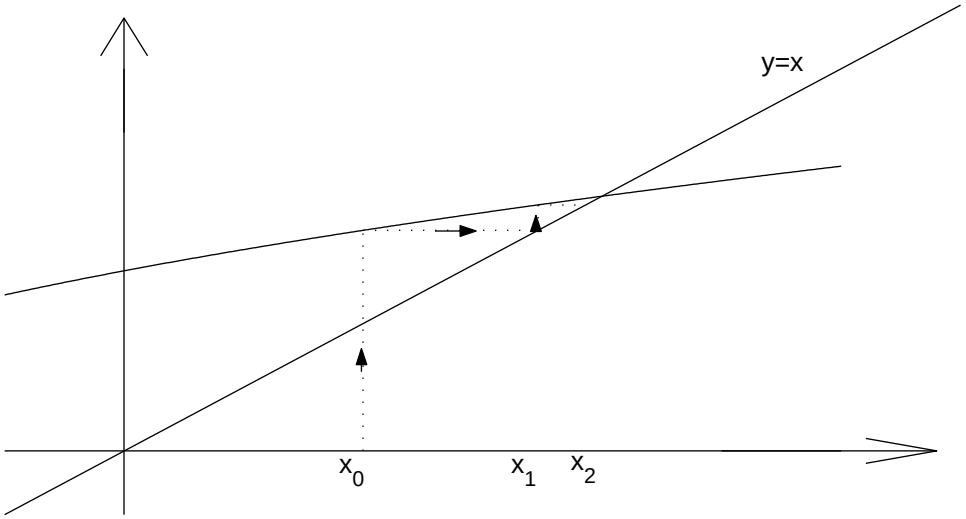

FIGURE 8.3 – Méthode de point fixe pour $g(x) = \sqrt{x+2}$

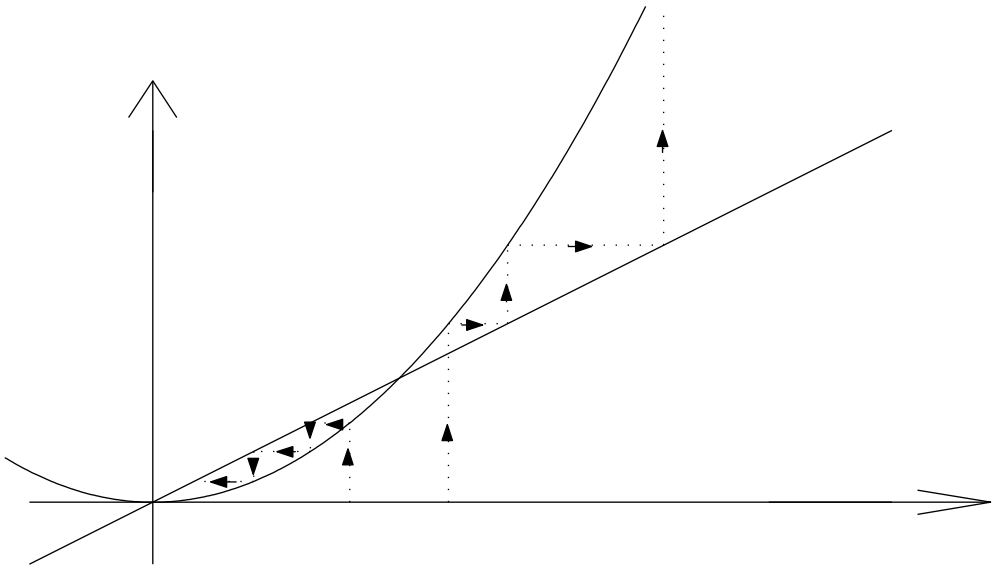

FIGURE 8.4 – Méthode de point fixe pour $g(x) = x^2$

Théorème 8.1 *Soit* $g : [a,b] \to [a,b]$ *dérivable et telle que*

$$\left|g'(x)\right| \leq K \ \ \forall x \in [a,b] \ \ avec \ 0 \leq K < 1 \tag{8.3}$$

Alors, pour tout $x_0 \in [a,b]$*, la suite définie par*

$$x_{n+1} = g(x_n) \ \ \forall n \in N$$

converge vers l'unique point fixe de g.

Démonstration : Notons d'abord que la suite est définie pout tout n puisque

$$\forall x \in [a,b] \qquad g(x) \in [a,b] \ .$$

La fonction g a un point fixe et un seul dans l'intervalle $[a,b]$, car la fonction $h(x) = g(x) - x$ vérifie :
(i) h est continue sur $[a,b]$
(ii) $h'(x) = g'(x) - 1 \leq k - 1 < 0$, donc h est strictement monotone
(iii) $h(a) = g(a) - a \geq 0$ puisque $g(a) \in [a,b]$
(iv) $h(b) = g(b) - b \geq 0$ puisque $g(b) \in [a,b]$.
Soit x_∞ ce point fixe, on a :

$$x_{n+1} - x_\infty = g(x_n) - g(x_\infty) = g'(\zeta_n)(x_n - x_\infty)$$

où $\zeta_n \in [a,b]$, d'après le théorème des accroissements finis. Ainsi, en utilisant (8.3), on a :

$$|x_{n+1} - x_\infty| \leq K |x_n - x_\infty| , \tag{8.4}$$

et par récurrence,

$$|x_n - x_\infty| \leq K^n |x_0 - x_\infty| \ \forall n.$$

Puisque $0 \leq K < 1$, ceci prouve que $\lim_{n\to\infty} |x_n - x_\infty| = 0$.

Remarque 8.5 Quand une suite x_n converge vers x_∞ selon la relation (8.4), on dit que la convergence est au moins linéaire. Plus généralement, on définit :

Définition 8.1 *Soit x_n une suite convergeant vers x_∞. S'il existe un nombre p et une constante $C \neq 0$ tels que*

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_\infty|}{|x_n - x_\infty|^p} = C,$$

on dit que la convergence est d'ordre p ; C est la constante d'erreur asymptotique.
S'il existe une constante C et un réel positif p tel que

$$|x_{n+1} - x_\infty| \leq C |x_n - x_\infty|^p$$

pour tout n assez grand, on dit que la convergence est au moins d'ordre p.

Remarque 8.6 La convergence vers 0 de $a_n - b_n$ dans l'algorithme 8.1 (la dichotomie) est linéaire.

Remarque 8.7 Il est clair qu'une convergence est d'autant plus rapide que son ordre est grand. En effet, si $|x_n - x_\infty|$ est petit, $|x_n - x_\infty|^2$ est encore plus petit !...

La notion de vitesse de convergence est évidemment essentielle en analyse numérique : afin de diminuer le temps de calcul (et donc le coût !), toutes choses étant égales par ailleurs, on choisira toujours l'algorithme convergeant le plus rapidement. Cependant, il est bien rare que "toutes les choses soient égales par ailleurs". Voir le paragraphe suivant à ce propos.

8.2.3.2 Méthode de Newton

Revenons un instant à la méthode du point fixe. D'après la formule de Taylor [1] à l'ordre 2, on a, en supposant g suffisamment régulière :

$$x_{n+1} - x_\infty = g(x_n) - g(x_\infty) = (x_n - x_\infty)g'(x_\infty) + \frac{1}{2}(x_n - x_\infty)^2 g''(\zeta_n)$$

où $\zeta_n \in [x_n, x_\infty]$. Pour n grand, on a donc

$$x_{n+1} - x_\infty \simeq (x_n - x_\infty)g'(x_\infty)$$

et la vitesse de convergence sera d'autant plus grande que $g'(x_\infty)$ est plus petit. Le cas le plus favorable est celui où $g'(x_\infty) = 0$. Si M_2 est majorant de $g''(x)$ sur $[a, b]$, on a alors :

$$|x_{n+1} - x_\infty| \leq \frac{M_2}{2} |x_n - x_\infty|^2 .$$

La convergence est alors au moins d'ordre 2 !

Si on revient à l'algorithme 8.3 de Newton, on voit qu'il s'agit en fait d'un algorithme de point fixe pour la fonction $g(x) = x - \frac{f(x)}{f'(x)}$. La dérivée de g est donnée par :

$$g'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x) f''(x)}{f'^2(x)} = \frac{f(x) f''(x)}{f'^2(x)}.$$

Si $f'(x_\infty) \neq 0$, on a alors, puisque $f(x_\infty) = 0$:

$$g'(x_\infty) = 0.$$

Ceci montre que la méthode de Newton converge de façon quadratique...si elle converge ! Pour s'assurer de sa convergence, on peut essayer d'appliquer à g le Théorème 8.1. Il est clair que l'hypothèse (8.3) est vérifiée sur un voisinage suffisamment petit de x_∞ (puisque $g'(x_\infty) = 0$ et on suppose g' continue). L'hypothèse plus difficile à vérifier est que g envoie un voisinage $[a, b]$ de x_∞ dans lui-même. On a en fait le résultat suivant :

Théorème 8.2 *Soit f deux fois continument différentiable sur un intervalle ouvert de centre x_∞ vérifiant :*

$$f(x_\infty) = 0, \ f'(x_\infty) \neq 0.$$

Alors, si x_0 est choisi assez près de x_∞, la méthode de Newton :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \ \forall n \geq 0$$

produit une suite x_n convergeant au moins quadratiquement vers x_∞.

Avant de démontrer ceci, faisons quelques remarques :

Remarque 8.8 Toute l'ambiguité dans ce résultat provient de "si x_0 est choisi assez près de x_∞" : en effet, dans la pratique, il n'y a généralement aucun moyen de savoir dans quelle mesure x_0 est assez voisin de x_∞.

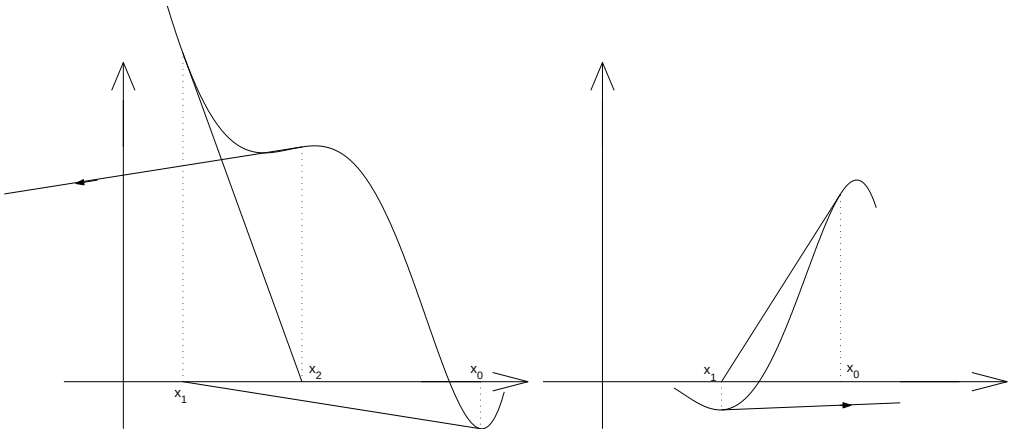

FIGURE 8.5 – Divergence dans la méthode de Newton.

C'est un des inconvénients de la méthode de Newton : une initialisation faite au hasard conduit souvent à une divergence. Ceci est suggéré par la Figure 8.5.

Remarque 8.9 L'hypothèse $f'(x_\infty) \neq 0$ est naturelle : étant donné le calcul du quotient $\frac{f(x_n)}{f'(x_n)}$, des valeurs petites de $f'(x_n)$ conduisent à une perte de précision. Si $f'(x_\infty) = 0$, il se peut cependant que la méthode converge, mais en général la convergence sera au plus linéaire.

Exemple : $f(x) = x^2$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = \frac{1}{2}x_n \ ;$$

ici x_n converge linéairement vers 0.

Remarque 8.10 Afin de lever l'incertitude de l'initialisation dans la méthode de Newton, on peut utiliser des théorèmes plus globaux comme celui assurant la convergence dès que $x_0 \in [a,b]$ si $f : [a,b] \to R$ est deux fois continuement différentiable et :
i) $f(a) f(b) \leq 0$
ii) $f'(x_\infty) \neq 0 \ \forall x \in [a,b]$
iii) $f''(x) \geq 0 \ (\leq 0) \ \forall x \in [a,b]$
iv) $\frac{|f(a)|}{|f'(a)|} < b-a, \ \frac{|f(b)|}{|f'(b)|} < b-a.$

Démonstration du Théorème 8.2 :
Nous allons montrer qu'il existe $\varepsilon > 0$ tel que la fonction

$$g(x) = x - \frac{f(x)}{f'(x)}$$

satisfasse les hypothèses du Théorème 8.1 dans l'intervalle $[x_\infty - \varepsilon, x_\infty + \varepsilon]$.
On a :

$$g'(x) = \frac{f(x) f''(x)}{f'^2(x)}$$

1. voir Section 1.7.1

Puisque $f'(x_\infty) \neq 0$, il existe ε_1 tel que :

$$\forall x \in [x_\infty - \varepsilon_1, x_\infty + \varepsilon_1],\ f'(x) \neq 0.$$

Il en résulte que g' est continue sur cet intervalle. Puisque $g'(x_\infty) = 0$, il existe $\varepsilon_2 \leq \varepsilon_1$ tel que :

$$\forall x \in [x_\infty - \varepsilon_2, x_\infty + \varepsilon_2],\ \left|g'(x)\right| \leq \frac{1}{2} < 1.$$

Par ailleurs, d'après le théorème des accroissements finis [2] :

$$|g(x) - g(x_\infty)| \leq |x - x_\infty| \max_{\zeta \in [x, x_\infty]} \left|g'(\zeta)\right| \leq \frac{1}{2}|x - x_\infty|,\ \text{si}\ |x - x_\infty| \leq \varepsilon_2.$$

puisque $g(x_\infty) = x_\infty$, ceci prouve que g envoie $[x_\infty - \varepsilon_2, x_\infty + \varepsilon_2]$ dans lui-même ($|x - x_\infty| \leq \varepsilon_2 \Longrightarrow |g(x) - x_\infty| \leq \varepsilon_2$). D'après le Théorème 8.1 appliqué à g avec $a = x_\infty - \varepsilon_2$, $b = x_\infty + \varepsilon_2$, la suite définie par

$$\begin{aligned} x_0 &\in [x_\infty - \varepsilon_2, x_\infty + \varepsilon_2] \\ x_{n+1} &= g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)} \end{aligned}$$

converge vers x_∞. Les considérations faites précédemment montrent que la convergence est quadratique, le seul point à vérifier étant le comportement de la dérivée seconde de g. Or, on vérifie que

$$g''(x_\infty) = \frac{f''(x_\infty)}{f'(x_\infty)}.$$

Si f est deux fois continument dérivable, g'' est bornée sur un voisinage de x_∞ et on peut directement appliquer les remarques précédentes.

8.2.3.3 Méthode de la sécante

L'algorithme 8.2 correspondant est défini par :

$$x_{n+1} := x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

soit encore

$$x_{n+1} - x_\infty = \frac{(x_{n-1} - x_\infty) f(x_n) - (x_n - x_\infty) f(x_{n-1})}{f(x_n) - f(x_{n-1})}$$

et, en utilisant $f(x_\infty) = 0$

$$x_{n+1} - x_\infty = (x_n - x_\infty)(x_{n-1} - x_\infty)\frac{\left[\frac{f(x_n) - f(x_\infty)}{x_n - x_\infty} - \frac{f(x_{n-1}) - f(x_\infty)}{x_{n-1} - x_\infty}\right]}{f(x_n) - f(x_{n-1})}.$$

2. voir Section 2.6.2

Utilisant les notations standard des différences divisées, voir aussi (2.6), (2.7) soit :

$$\begin{aligned} f[x] &:= f(x) \\ f[x,y] &:= \frac{f(y)-f(x)}{y-x} = \frac{f[y]-f[x]}{y-x} \\ f[a,x,y] &:= \frac{f[x,y]-f[a,x]}{y-a}, \text{ etc...,} \end{aligned}$$

la formule ci-dessus s'écrit :

$$x_{n+1} - x_\infty = (x_n - x_\infty)(x_{n-1} - x_\infty) \frac{f[x_{n-1}, x_\infty, x_n]}{f[x_{n-1}, x_n]}. \tag{8.5}$$

D'après le théorème des accroissements finis [3]

$$f[x_{n-1}, x_n] = f'(\zeta_n) \text{ où } \zeta_n \in [x_{n-1}, x_n]. \tag{8.6}$$

Nous avons vu au Lemme 2.1 que :

$$f[x_{n-1}, x_\infty, x_n] = \frac{1}{2} f''(\eta_n) \text{ où } \eta_n \in \text{ plus petit intervalle contenant } x_{n-1}, x_n, x_\infty. \tag{8.7}$$

Si on suppose que x_n prend ses valeurs dans un intervalle J sur lequel :

$$\forall x \in J \ \left|f''(x)\right| \leq M, \ \left|f'(x)\right| \geq m > 0, \tag{8.8}$$

d'après (8.5), (8.6), (8.7) et en posant :

$$e_n := |x_n - x_\infty|,$$

on a :

$$e_{n+1} = c_n e_n e_{n-1}, \quad c_n \leq \frac{1}{2}\frac{M}{m} \tag{8.9}$$

(et $\lim_{n\to\infty} c_n = \frac{1}{2}\frac{f''(x_\infty)}{f'(x_\infty)}$ d'après ce qui suit). Si f est deux fois continument dérivable sur un intervalle $[x_\infty - \alpha, x_\infty + \alpha]$ et si $f'(x_\infty) \neq 0$, on montre, comme dans le cas de la méthode de Newton qu'il existe $J = [x_\infty - \varepsilon, x_\infty + \varepsilon]$ $(0 < \varepsilon \leq \alpha)$ et M, $m > 0$ tel que (8.8) soit vérifié. Quitte à resteindre encore ε, en utilisant (8.9), on montre que si $x_0 \in J$, alors x_n appartient à J pour tout n, ceci par récurrence. Dans ce cas, l'estimation (8.9) est valable pour tout n. On déduit de cette estimation que e_n tend vers 0 et que la convergence est surlinéaire. En effet, remarquons d'abord qu'on peut se "ramener" au cas $C = 1$ dans (8.9) en posant $\varepsilon_n = Ce_n$. Multipliant alors (8.9) par C, on a

$$\varepsilon_{n+1} \leq (Ce_n)(Ce_{n-1}) = \varepsilon_n \varepsilon_{n-1}. \tag{8.10}$$

Soit alors p la racine positive de $p^2 - p - 1 = 0$, soit $p = \frac{(1+\sqrt{5})}{2} = 1,618...$ (le nombre d'or !) et $K = \max\left(\varepsilon_0, \sqrt[p]{\varepsilon_1}\right)$. Montrons par récurrence que

$$\varepsilon_n \leq K^{p^n}. \tag{8.11}$$

3. voir Section 2.6.2

Ceci est vrai pour $n = 0, 1$ d'après le choix de K. Si c'est vrai jusqu'au rang n, d'après (8.10), on a :

$$\varepsilon_{n+1} \leq K^{p^n} K^{p^{n-1}} = K^{p^{n-1}(1+p)} = K^{p^{n+1}}, \text{ d'après le choix de } p\,;$$

donc (8.10) est vrai pour tout n. Ainsi, si on choisit $K < 1$ (c'est-à-dire x_0 et x_1 suffisamment voisins de x_∞), ε_n et donc e_n tendent vers 0 quand n tend vers l'infini, d'après (8.11). Cette même relation (8.11) suggère que la convergence est au moins d'ordre p. On le montre (cf. Définition 8.1), en remarquant que (8.10) implique :

$$\frac{\varepsilon_{n+1}}{\varepsilon_n^p} \leq \varepsilon_n^{1-p} \varepsilon_{n-1} = \left(\frac{\varepsilon_n}{\varepsilon_{n-1}^p} \right)^{1-p} \text{ car } p\,(1-p) = -1.$$

Ainsi $\frac{\varepsilon_n}{\varepsilon_{n-1}^p}$ reste inférieur à la suite Y_n définie par

$$\begin{aligned} Y_1 &= \frac{\varepsilon_1}{\varepsilon_0^p}, \\ Y_{n+1} &= Y_n^{1-p} \end{aligned}$$

qui, comme on le vérifie aisément converge vers 1 puisque $|1-p| < 1$.
Donc, pour n assez grand

$$\varepsilon_{n+1} \leq Y_n \varepsilon_n^p \leq (1+\varepsilon)\, \varepsilon_n^p.$$

Ceci prouve que la convergence est au moins d'ordre $p = 1,618...$.

Théorème 8.3 *Soit f deux fois continument différentiable sur un intervalle ouvert de centre x_∞ vérifiant :*

$$f(x_\infty) = 0,\ f'(x_\infty) \neq 0.$$

Alors, si x_0, x_1 sont choisis assez près de x_∞, l'algorithme de la sécante défini par :

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \ \forall n \geq 1$$

converge vers x_∞ et la convergence est au moins d'ordre $p = 1,618...$.

8.2.4 Comparaison des algorithmes

Pour comparer les algorithmes, il est important de tenir compte de la présence ou non des facteurs suivants :
- assurance de la convergence
- vitesse de la convergence
- stabilité de l'algorithme - précision des résultats
- efficacité des calculs (ex : nombre de fonctions à calculer à chaque itération).

8.2.4.1 Méthode de dichotomie

Avantages :
- la convergence est assurée
- on a un encadrement de la solution

– un seul calcul de fonction à chaque itération.

Inconvénients :

– vitesse de convergence linéaire, donc lente

– sensible aux erreurs d'arrondi ; exemple : la fonction $f(x) = e^x - 1 - x - \frac{x^2}{2}$ s'annule théoriquement en $x = 0$ seulement. Numériquement, elle change de signe un grand nombre de fois autour de $x = 0$.

8.2.4.2 Méthode de Newton

Avantages :

– converge rapidement quand elle converge (ce qui compense largement le dernier inconvénient)

– relativement stable et peu sensible aux erreurs d'arrondis si $f'(x_\infty)$ n'est pas trop petit.

Inconvénients :

– peut diverger ou converger vers un autre zéro que celui cherché si la donnée initiale est mal choisie

– nécessite le calcul de la dérivée d'une fonction, ce qui est numériquement difficile si on ne la connait pas explicitement

– chaque étape nécessite deux évaluations de fonctions.

8.2.4.3 Méthode de la sécante

Avantages :

– nécessite une seule évaluation de fonction à chaque étape

– convergence relativement rapide, bien que moins que celle de Newton.

Inconvénients :

– comme la méthode de Newton, peut diverger si les données initiales sont mal choisies

– le calcul de $f(x_n) - f(x_{n-1})$ peut produire des erreurs de chute.

8.2.4.4 Méthode du point fixe

Inconvénients :

– convergence souvent difficile à réaliser en pratique

– certains points fixes -dits instables- ne peuvent être atteints

– convergence lente de type linéaire, en général.

Conclusion : cette dernière méthode a surtout un interêt théorique : son analyse mathématique est relativement aisée et nous a permis d'en déduire celle de la méthode de Newton qui en est en fait un cas particulier. En pratique, elle est souvent utilisée dans de tels cas particuliers. Nous verrons à plusieurs reprises dans la suite qu'elle joue un rôle considérable pour adapter des algorithmes linéaires à des situations non linéaires. Elle est donc fondamentale en tant que concept. D'autre part, couplée avec des techniques d'accélération de la convergence (cf. paragraphe suivant), elle peut être intéressante.

La méthode de Newton a généralement la faveur des utilisateurs : elle est effectivement très rapide et est certainement à conseiller lorsque le calcul de f' est aisé et lorsque des remarques simples permettent de choisir à coup sûr la donnée initiale dans le domaine de convergence. Pour cela, elle peut être, par exemple, précédée d'une application de quelques itérations de la méthode de dichotomie.

Si on veut une méthode d'application plus universelle, la méthode de la sécante est plutôt à conseiller puisqu'elle nécessite seulement la connaissance de f. Elle est relativement rapide et ne nécessite qu'une évaluation de fonction à chaque itération. Ce dernier point peut même la faire considérer comme plus rapide que la méthode de Newton. En effet, si on pose :

$$u_n := x_{2n},$$

le passage de u_n à u_{n+1} nécessite le même travail qu'une itération de la méthode de Newton. Mais on a :

$$|u_{n+1} - x_\infty| \leq C\,|x_{2n+1} - x_\infty|^p \leq C\,|u_n - x_\infty|^{p^2}.$$

Vue ainsi, la méthode est d'ordre $p^2 = 2,618...$! Ce genre de considérations n'est pas à négliger dans une évaluation globale du temps de calcul.

Terminons par un exemple :

Résolution de $x - 0,2\sin x - 0,5 = 0$ à l'aide des quatre algorithmes différents

	Dichotomie $x_{-1} = 0,5$ $x_0 = 1,0$	Sécante $x_{-1} = 0,5$ $x_0 = 1,0$	Newton $x_0 = 1$	Point fixe $x_0 = 1$ $x = 0,2\sin x + 0,5$
1	0,75	0,5	0,5	0,50
2	0,625	0,61212248	0,61629718	0,595885
3	0,5625	0,61549349	0,61546820	0,612248
4	0,59375	0,61546816	0,61546816	0,614941
5	0,609375			0,61538219
6	0,6171875			0,61545412
7	0,6132812			0,61546587
8	0,6152343			0,61546779
9	0,6162109			0,61546810
10	0,6157226			0,61546815
11	0,6154785			
12	0,6153564			
13	0,6154174			
14	0,6154479			
15	0,6154532			
16	0,61547088			
17	0,61546707			
18	0,61546897			
19	0,615468025			
20	0,615468502			

Cet exemple confirme les remarques générales. La méthode de Newton est la plus rapide. Ici, la méthode de la sécante l'est presque autant. Les deux autres sont de type linéaire : celle du point fixe est plus rapide ce qui est cohérent avec le fait que son taux de convergence est

$$\max_x \left| \frac{d}{dx}\left(0.2\sin x + 0.5\right) \right| = 0.2$$

au lieu de 0.5 pour la première.

8.2.5 Accélération de la convergence

D'une manière générale, étant donnée une suite x_n convergeant vers x_∞, accélérer sa convergence consisite à la remplacer par une suite $\widehat{x}_n$ convergeant plus vite vers x_∞, c'est-à-dire vérifiant :

$$\lim_{n\to\infty} \frac{\widehat{x}_n - x_\infty}{x_n - x_\infty} = 0.$$

Par exemple, si x_n converge linéairement, on cherchera à construire $\widehat{x}_n$ convergeant quadratiquement.

8.2.5.1 Méthode de relaxation

Elle est basée sur une idée très simple. Considérons une méthode de point fixe $x_{n+1} = g(x_n)$ qui ne converge pas ou très lentement vers un point fixe x^*. L'équation qu'on cherche à résoudre $x = g(x)$ peut également s'écrire

$$\alpha x + x = g(x) + \alpha x \tag{8.12}$$

et ce pour tout α réel. L'écriture de l'équation sous la forme (8.12) suggère d'utiliser une nouvelle méthode de point fixe :

$$x_{n+1} = \frac{g(x_n) + \alpha x_n}{\alpha + 1} := G(x_n). \tag{8.13}$$

Maintenant, d'après le Théorème 8.1, cette méthode (8.13) va converger (en choisissant d'initialiser assez près de la solution) dès que

$$|G'(x^*)| = \left| \frac{g'(x^*) + \alpha}{\alpha + 1} \right| < 1$$

et ce d'autant plus rapidement que $\left| \frac{g'(x^*) + \alpha}{\alpha + 1} \right|$ est petit. Comme on est parfaitement libre de choisir le paramètre α (qui s'appelle ici **paramètre de relaxation**), l'idée est de le prendre le plus proche possible de $-g'(x^*)$. Bien sûr, la valeur exacte de $g'(x^*)$ n'est en général pas connue, mais par encadrement on peut en avoir des valeurs aprochées convenables, qu'on peut d'ailleurs réactualiser éventuellement au cours des itérations. En conclusion cette méthode est très simple d'utilisation et peut rendre de précieux services.

8.2.5.2 Méthode du Δ^2 d'Aitken

Afin d'illustrer la méthode, commençons par considérer une suite x_n convergeant vers x_∞ et telle que :

$$x_{n+1} - x_\infty = k\,(x_n - x_\infty) \text{ où } 0 < k < 1.$$

Les nombres k et x_∞ peuvent être exprimés en fonction des valeurs de la suite x_n à l'aide des deux équations :

$$\begin{aligned} x_{n+1} - x_\infty &= k\,(x_n - x_\infty)\,, \\ x_{n+2} - x_\infty &= k\,(x_{n+1} - x_\infty)\,. \end{aligned}$$

On obtient par différence :

$$x_{n+2} - x_{n+1} = k\,(x_{n+1} - x_n)\,,$$

et en substituant dans la première équation mise sous la forme

$$x_\infty = x_n + \frac{x_{n+1} - x_n}{1 - k}$$

on a :

$$x_\infty = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} + x_n - 2x_{n+1}}.$$

En utilisant les notations des différences finies, soit

$$\begin{aligned} \Delta x_n &= x_{n+1} - x_n \\ \Delta^2 x_n &= \Delta x_{n+1} - \Delta x_n = (x_{n+2} - x_{n+1}) - (x_{n+1} - x_n), \end{aligned}$$

ceci s'écrit encore :

$$x_\infty = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}.$$

La méthode tire son nom de cette formule. Elle montre ici que la limite x_∞ cherchée peut être exactement connue à l'aide de x_{n+2}, x_{n+1}, et x_n ; n quelconque.
L'idée d'Aitken est de généraliser cette remarque aux suites convergeant linéairement c'est-à-dire telles que :

$$x_{n+1} - x_\infty = k_n (x_n - x_\infty)$$

avec

$$\lim_{n\to\infty} k_n = k \in [0, 1[.$$

Pour une telle suite, le coefficient k_n est "presque" constant si n est grand et on peut alors faire le calcul précédent pour se convaincre que :

$$\widehat{x}_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$$

doit être très voisin de x_∞. On a en effet :

Théorème 8.4 *Soit une suite x_n vérifiant pour tout entier n, $x_n \neq x_\infty$ et $x_{n+1} - x_\infty = (k + \varepsilon_n)(x_n - x_\infty)$ avec $0 \leq |k| < 1$ et $\lim_{n\to\infty} \varepsilon_n = 0$. Alors la suite définie par :*

$$\widehat{x}_n := x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$$

vérifie :

$$\lim_{n\to\infty} \frac{\widehat{x}_n - x_\infty}{x_n - x_\infty} = 0.$$

Démonstration : Posons $e_n = x_n - x_\infty$. On a :

$$\begin{aligned} \Delta^2 x_n &= e_{n+2} - 2e_{n+1} + e_n = e_n \left[(k + \varepsilon_{n+1})(k + \varepsilon_n) - 2(k + \varepsilon_n) + 1\right] \\ &= e_n \left[(k-1)^2 + \eta_n\right] \text{ où } \eta_n = k(\varepsilon_{n+1} + \varepsilon_n) + \varepsilon_{n+1}\varepsilon_n - 2\varepsilon_n. \end{aligned}$$

Puisque η_n tend vers 0 et $e_n \neq 0$, $\Delta^2 x_n$ est différent de 0 pour n grand et $\widehat{x}_n$ est donc bien défini au moins pour n assez grand. D'autre part

$$\widehat{x}_n - x_\infty = e_n \left[1 - \frac{(k - 1 + \varepsilon_n)^2}{(k-1)^2 + \eta_n}\right].$$

Mais l'expression entre crochets tend vers 0 quand n tend vers l'infini ce qui prouve le résultat.

Application : Algorithme de Steffensen Soit x_n définie par :

$$x_{n+1} = g(x_n) \ \forall n \geq 0$$

où g et x_0 vérifient les hypothèses du théorème 8.1. On sait qu'alors x_n converge au moins linéairement vers x_∞. On peut accélérer cette convergence en utilisant la méthode de Δ^2 d'Aitken. Il est alors plus efficace de repartir avec la nouvelle valeur $\widehat{x}_n$ à chaque nouvelle itération. Ceci donne l'algorithme suivant dit de Steffensen.

Algorithme 8.5 x_0 donné

$$\left[\begin{array}{c} \text{Pour } n = 0, 1, 2, ... \\ y_n := g(x_n), \ z_n := g(y_n) \\ x_{n+1} := x_n - \dfrac{(y_n - x_n)^2}{z_n - 2y_n + x_n} \end{array}\right.$$

Remarque 8.11 Cet algorithme est à nouveau un algorithme de point fixe

$$\begin{aligned} x_{n+1} &= \psi(x_n) \text{ avec} \\ \psi(x) &= x - \frac{(g(x) - x)^2}{g(g(x)) - 2g(x) + x}. \end{aligned}$$

On peut montrer que si $g'(x_\infty) \neq 0$, alors $\psi'(x_\infty) = 0$. On sait qu'alors l'algorithme associé à ψ converge de façon quadratique. Ainsi, par rapport à l'algorithme associé à g :
– on accélère la convergence quand elle a lieu
– on a un procédé convergeant, même si $|g'(x_\infty)| \geq 1$.

Exemple 8.5 Nous avons vu que si $g(x) = x^2$, la suite définie par

$$x_{n+1} = x_n^2$$

converge si $|x_0| < 1$ vers le point fixe 0. Le point fixe 1 n'est jamais atteint (sauf dans le cas irréaliste où x_0 est exactement égal à 1). Ceci est cohérent avec le fait que $g'(1) = 2 > 1$. Calculons la fonction ψ dans ce cas :

$$\psi(x) = x - \frac{\left(x^2 - x\right)^2}{x^4 - 2x^2 + x} = x - \frac{x^2(x-1)^2}{x(x-1)(x^2 + x - 1)}$$

$$\psi(x) = \frac{x\left(x^2 + x - 1\right) - x(x-1)}{x^2 + x + 1} = \frac{x^3}{x^2 + x - 1} = \frac{x^3}{(x - r_1)(x - r_2)}$$

où $r_{1,2} = \dfrac{-1 \pm \sqrt{5}}{2}$.
Puisque $\psi(x) \sim x^3$ pour x voisin de 0, si x_0 est choisi assez petit, la suite x_n se comportera sensiblement comme :

$$Y_{n+1} = Y_n^3,$$

d'où convergence d'ordre 3 vers 0.
Si x_0 est choisi voisin de 1, puisque $\psi'(1) = 0$ et $\psi''(1) \neq 0$, la suite associée à ψ converge de façon quadratique vers 1. Ici, il y a donc un gain considérable par rapport à g.

Remarque 8.12 Il faut noter que, si l'algorithme 8.5 converge plus vite que celui associé à g, chaque itération comporte deux évaluations de la fonction : on y met le prix. Ainsi, une comparaison plus juste consiste à mettre sur le même plan la suite x_n définie par l'algorithme 8.5 et la suite Y_n définie par :

$$Y_{n+1} = g\left(g\left(Y_n\right)\right),\ Y_0 = 1$$

où chaque itération comporte aussi deux évaluations de g. Dans l'exemple précédent, on a $Y_{n+1} = Y_n^4$, qui, au voisinage de 0, est d'ordre 4 et donc meilleure que celle donnée par ψ qui est d'ordre 3. Bien sûr, on a dans ce cas $g'\left(x_\infty\right) = 0$!

Remarque 8.13 D'une manière générale, il est illusoire d'espérer obtenir des méthodes convergeant plus rapidement sans alourdir le travail à chaque itération, soit en augmentant le nombre d'évaluation de fonctions, soit en évaluant plus de dérivées. Signalons par exemple l'algorithme de Richmond d'ordre 3 pour résoudre $f\left(x\right) = x$ donné par :

$$x_{n+1} = x_n - \frac{2f\left(x_n\right)f'\left(x_n\right)}{f'\left(x_n\right)^2 - f\left(x_n\right)f''\left(x_n\right)}.$$

Cet algorithme parfois utilisé converge bien sûr rapidement, mais nécessite l'évaluation de $f,\ f',\ f''$ à chaque étape.

Théoriquement, il est presque toujours possible d'imaginer des méthodes d'ordre aussi élevé qu'on veut. Du point de vue pratique, elles sont le plus souvent irréalistes. D'autre part, il faut se convaincre qu'une méthode d'ordre 2 est déjà très rapide et qu'en général, il est pratiquement difficile de faire mieux. On peut se reporter à l'exemple traité précédemment : en 4 itérations, la méthode de Newton donne le résultat avec 8 chiffres significatifs ! Un numéricien raisonnable s'en contentera.

8.3 Systèmes d'équations non linéaires

Reprenons les méthodes étudiées au paragraphe 8.2. Il est clair que la méthode de dichotomie n'a aucun sens en dimension $N \geq 2$. En revanche, les trois autres méthodes peuvent s'adapter au cas d'une fonction $F : \mathbb{R}^N \longrightarrow \mathbb{R}^N$. Passons rapidement sur les méthodes de point fixe qui nécessitent que le système d'équations à résoudre soit écrit sous la forme

$$\left\{\begin{array}{rcl} x_1 & = & g_1(x_1, x_2, \ldots, x_N) \\ x_2 & = & g_2(x_1, x_2, \ldots, x_N) \\ \vdots & & \vdots \\ x_N & = & g_N(x_1, x_2, \ldots, x_N)\ . \end{array}\right. \tag{8.14}$$

Il est assez rare que la méthode de point fixe écrite à partir du système (8.14) converge, les conditions de convergence étant plus difficiles à réaliser qu'en dimension un. Il faut en effet que pour une certaine norme matricielle (voir chapitre 6), la norme de la matrice Jacobienne de F soit strictement inférieure à 1 au point recherché. On peut bien sûr envisager des méthodes d'accélération de convergence comme au paragraphe 8.2.5. Nous faisons le choix de présenter plutot ici la méthode généralisant la méthode de Newton.

8.3.1 La méthode de Newton-Raphson

Considérons un système d'équations s'écrivant $F(X) = 0$ ou encore, de façon développée

$$\left\{ \begin{array}{ll} f_1(x_1, x_2, \ldots, x_N) & = 0 \\ f_2(x_1, x_2, \ldots, x_N) & = 0 \\ \vdots & \vdots \\ f_N(x_1, x_2, \ldots, x_N) & = 0 \ . \end{array} \right. \tag{8.15}$$

La généralisation naturelle de la formule de Newton unidimensionnelle

$$x_{n+1} = x_n - (f'(x_n))^{-1} f(x_n)$$

fait intervenir la matrice Jacobienne de F :

$$F'(X_n) = \left(\begin{array}{cccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{array} \right) \tag{8.16}$$

toutes les dérivées partielles étant évaluées au point X_n. La méthode de Newton-Raphson s'écrit donc formellement

$$X_{n+1} = X_n - [F'(X_n)]^{-1} F(X_n) \ . \tag{8.17}$$

Le second membre de (8.17) est parfaitement défini car on effectue le produit d'une matrice $N \times N$ par un vecteur de $\mathbb{R}^N$. Dans la pratique, on ne calcule pas explicitement l'inverse de la matrice Jacobienne, ce qui s'avèrerait trop coûteux, et on préfère écrire l'algorithme sous la forme suivante :

Algorithme 8.6 (Newton-Raphson)

$$\left[\begin{array}{l} X_0 \text{ donné , pour } n = 0, 1, \ldots, \text{ test d'arrêt , faire} \\ \text{Résolution du système linéaire } F'(X_n)\delta_n = -F(X_n) \\ X_{n+1} = X_n + \delta_n \end{array} \right.$$

Remarque 8.14 Encore plus qu'en dimension 1, le choix de l'initialisation est crucial et le risque de divergence, si on ne démarre pas à proximité de la solution cherchée, est grand.

Remarque 8.15 La convergence est là aussi d'ordre 2, donc très rapide (quand il y a convergence !)

Remarque 8.16 Dans le cas des systèmes de grande dimension, la méthode de Newton-Raphson s'avère assez coûteuse puisqu'il faut évaluer à chaque itération $N^2 + N$ fonctions (les N^2 dérivées partielles de la matrice Jacobienne, plus les N fonctions coordonnées). De plus, il y a un système linéaire $N \times N$ (dont la matrice est en général pleine) à résoudre. Pour pallier le premier inconvénient, il est fréquent qu'on ne recalcule pas la matrice Jacobienne à chaque itération, mais seulement de temps en temps. Bien sûr, à ce moment-là, la convergence n'est plus quadratique, mais en temps de calcul il arrive qu'on y gagne beaucoup, en particulier quand les dérivées de F sont très coûteuses à calculer.

8.3.2 Méthodes de type quasi-Newton

Il existe des méthodes pour les systèmes, dans l'esprit de la méthode de la sécante, qui approchent en un certain sens la résolution du système linéaire apparaissant à chaque étape. Ces méthodes sont appelées méthodes quasi-Newton. Citons par exemple la méthode de Broyden. Son principe est le suivant. A la i-ème itération, on va chercher à approcher la matrice Jacobienne de F qui intervient dans l'algorithme de Newton-Raphson, par une matrice B_i. Pour être consistant, on va imposant que B_i satisfasse

$$B_i(X_i - X_{i-1}) = F(X_i) - F(X_{i-1})\,. \tag{8.18}$$

Alors, on chosira X_{i+1} tel que $B_i(X_{i+1} - X_i) = -F(X_i)$ comme dans la formule de Newton.

Cette condition (8.18) ne définit évidemment pas B_i de façon unique, car elle n'impose une condition que dans une direction.L'idée de Broyden est d'actualiser les matrices B_i en faisant le moins de changements possibles, ce qui peut être réalisé en rajoutant juste une matrice de rang 1 à B_i pour obtenir B_{i+1}. Plus précisément, en notant $\delta X_i = X_{i+1} - X_i$ et $\delta F_i = F(X_{i+1}) - F(X_i$, on obtient B_{i+1} à partir de B_i par la formule

$$B_{i+1} = B_i + \frac{(\delta F_i - B_i.\delta X_i) \bigotimes \delta X_i}{\delta X_i.\delta X_i} \tag{8.19}$$

où le produit tensoriel de deux vecteurs U et V est la matrice UV^t de terme général $u_i v_j$.

8.4 Exercices du chapitre 8

Exercice 8.1 Etudier la convergence des méthodes de point fixe suivantes (on cherche à approcher $\sqrt{2}$). Appliquer des méthodes d'accélération de la convergence pour rendre ces algorithmes convergents quand ils ne convergent pas ou plus rapides quand ils convergent.

$$\begin{aligned}
x_{n+1} &= \tfrac{2}{x_n} \\
x_{n+1} &= \tfrac{1}{2}\,(x_n + \tfrac{2}{x_n}\,) \\
x_{n+1} &= 2x_n - \tfrac{2}{x_n} \\
x_{n+1} &= \tfrac{1}{3}\,(2x_n + \tfrac{2}{x_n}\,)
\end{aligned}$$

Exercice 8.2 Pour les fonctions suivantes, trouver un algorithme de point fixe (autre que la méthode de Newton !) qui converge vers le plus petit zéro positif de

1. $x^3 - x - 1 = 0$
2. $x - \tan x = 0$
3. $e^{-x} - \cos x = 0$.

Exercice 8.3 Trouver les racines des polynômes suivants

$$x^4 - 56,101x^3 + 785,6561x^2 - 72,7856x + 0,078$$

$$x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - Ax^2 + 13068x - 5040$$

avec $A = 13132$, puis $A = 13133$.

Exercice 8.4 En utilisant la formule de Taylor, imaginer un algorithme modifiant la méthode de Newton et assurant une convergence quadratique dans le cas d'une racine multiple de f.

Exercice 8.5 *Vérifier que la suite de matrices B_i définies par (8.19) vérifie bien la condition (8.18).*

8.5 TD8 : recherche de zéros de fonctions à une ou plusieurs variables

Exercice 1 : Recherche d'un zéro. On souhaite résoudre l'équation non linéaire $f(x) = 0$ sur $]0; +\infty]$, où f est définie par

$$f(x) := x^2 - \sin(x). \tag{8.20}$$

1. Donner graphiquement une approximation du zéro strictement positif de f.
2. Imaginer et programmer une méthode de point fixe. Converge-t-elle ? En combien d'itérations ?
3. Maintenant, programmez la méthode de Newton. Que remarquez-vous au niveau de la convergence ? Jouez avec la valeur de x_0, initialisation de Newton. Converge-t-elle toujours ?
4. Programmer maintenant la méthode de la sécante. Comparer son comportement aux deux précédentes méthodes. Quelle conclusion tirez-vous de cela ?

Exercice 2
Soit F la fonction suivante :

$$F : \begin{array}{c} \mathbb{R}^3 \longrightarrow \mathbb{R}^3 \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \longmapsto \begin{pmatrix} 40x^3 - 12xz^2 - 16xy + 50x - 4y + 2z \\ -8x^2 - 4x + 50y - 4z - 10 \\ 4z^3 - 12x^2z + 2x - 4y - 8z - 24 \end{pmatrix} \end{array}$$

Le but de cet exercice est de trouver un zéro de F, par trois méthodes :
1. Soit ϕ définie par :

$$\phi(x, y, z) = f_1(x, y, z)^2 + f_2(x, y, z)^2 + f_3(x, y, z)^2$$

Où f_1, f_2, f_3 sont les fonctions composantes de F. Rechercher à l'aide de la fonction Matlab *fminsearch* un minimum de ϕ. Est-ce un zéro de F ?
2. Programmer la méthode de Newton pour F (il faudra calculer la matrice jacobienne de f).
3. Utiliser la fonction *fsolve* de Matlab.

Chapitre 9

Eléments sur l'optimisation : aspects théoriques et numériques

9.1 Aspects théoriques

9.1.1 Introduction

On s'intéresse dans ce chapitre au problème du type suivant : "trouver le minimum d'une fonction avec ou sans contrainte(s)". D'un point de vue mathématique, le problème se formule de la façon suivante :

$$\min_{\mathbf{x}\in\mathbb{R}^n} J(\mathbf{x}).$$

L'espace sur lequel s'effectue la minimisation peut être un sous-espace C de $\mathbb{R}^n$ ce qui permet de considérer des contraintes. Le problème peut être également posé en dimension infinie. il s'agit dans ce cas de minimiser une fonctionnelle J sur un espace vectoriel de dimension *infinie* (un espace hilbertien par exemple mais non exclusivement). Toutefois, nous adopterons, dans ce cours élémentaire, le point de vue qui consiste à traiter des problèmes d'optimisation en dimension *finie*. La raison principale est, qu'en pratique, on discrétise un problème continu, ce qui consiste à faire une projection sur un espace de dimension finie. En plus d'introduire les notations et notions de l'optimisation, nous donnons également quelques résultats théoriques sur l'optimisation sans ou avec contraintes. Remarquons que
– maximiser J est équivalent à minimiser $-J$,
– il est important de distinguer entre minimum local et global.

9.1.2 Résultats d'existence

La plupart des théorèmes d'existence de minimum sont des variantes du théorème classique suivant : une fonction continue sur un compact admet un minimum. Commençons par le théorème suivant.

Théorème 9.1 *Soit J une fonction continue sur un sous-ensemble C fermé de $\mathbb{R}^n$. On suppose que*
– *ou bien C est borné,*
– *ou bien C est non borné et $\lim_{\|\mathbf{x}\|\to+\infty} J(\mathbf{x}) = +\infty$ (on dit alors que J est coercive),*

alors J possède un minimum sur C.

Preuve.
si i) a lieu C est compact et la conclusion est alors évidente.
Si ii) a lieu on peut par exemple prouver le résultat comme suit. Soit (x_n) une suite minimisante pour J, c'est à dire une suite telle que

$$J(x_n) \underset{n\to\infty}{\to} \inf_{x\in C} J(x). \tag{9.1}$$

Puisque J est coercive, on vérifie que (x_n) est bornée et on peut donc en extraire une sous suite (x_{n_k}) qui converge vers un certain élément x^*. Puisque J est continue, $J(x_{n_k})$ converge vers $J(x^*)$. On déduit alors de (9.1) que $J(x^*) = \inf_{x\in C} J(x)$. □

9.1.3 Convexité

La convexité joue un rôle extrêmement important en optimisation. Donnons quelques définitions.

Définition 9.1 *Un ensemble C est dit convexe si, pour tous points $\mathbf{x}$ et $\mathbf{y}$ de C, le segment $[\mathbf{x};\mathbf{y}]$ est inclus dans C, i.e., $\forall t \in [0;1]$, $t\mathbf{x} + (1-t)\mathbf{y}$ est un point de C. Une fonction J définie sur un ensemble convexe C est dite convexe si*

$$\forall (\mathbf{x},\mathbf{y}) \in C\times C, \quad \forall t \in [0;1], \quad J(t\mathbf{x} + (1-t)\mathbf{y}) \leq tJ(\mathbf{x}) + (1-t)J(\mathbf{y}).$$

La fonction est dite strictement convexe si

$$\forall (\mathbf{x},\mathbf{y}) \in C\times C, \quad \mathbf{x}\neq\mathbf{y}, \quad \forall t \in]0;1[, \quad J(t\mathbf{x} + (1-t)\mathbf{y}) < tJ(\mathbf{x}) + (1-t)J(\mathbf{y}).$$

Lorsqu'une fonction convexe est dérivable, la caractérisation suivante sera utile.

Proposition 9.1 *Soit J une fonction différentiable définie sur un convexe C de $\mathbb{R}^n$, alors J est convexe si et seulement si*

$$\forall (\mathbf{x},\mathbf{y}) \in C\times C, \quad \nabla J(\mathbf{x})\cdot(\mathbf{y}-\mathbf{x}) \leq J(\mathbf{y}) - J(\mathbf{x}).$$

Preuve.
• Soit J convexe. On considère $\theta \in [0,1]$, et $x, y \in C$. Puisque C est convexe $x+\theta(y-x) \in C$ et en utilisant la convexité de J il vient : $J(x+\theta(y-x)) \leq (1-\theta)J(x) + \theta J(y)$. D'où :

$$J(y) - J(x) \geq \frac{J(x+\theta(y-x)) - J(x)}{\theta}. \tag{9.2}$$

On conclut alors en faisant tendre θ vers 0 dans (9.2).
• Réciproquement, on part des deux inégalités :

$$\begin{aligned} J(y) &\geq J(y+\theta(x-y)) - \theta J(y+\theta(x-y))(x-y) \\ J(x) &\geq J(y+\theta(x-y)) + (1-\theta)J(y+\theta(x-y))(x-y). \end{aligned}$$

En combinant ces deux inégalités on obtient : $J(x+(1-\theta)y) \leq \theta J(x) + (1-\theta)J(y)$, ce qui établit la convexité. □

Remarque 9.1 *On a une caractérisation similaire pour les fonctions strictement convexe : remplacer convexe par strictement convexe et l'inégalité large par une inégalité stricte dans la proposition 9.1. Toutefois le premier sens de la preuve ne peut pas être directement adapté. En effet, si on l'obtient bien (9.2) avec une inégalité stricte, cette dernière n'est pas nécéssairement conservée lorsque θ tend vers 0.*

Le résultat suivant montre l'impact de la convexité dans les problèmes d'optimisation.

Proposition 9.2 *Soit J une fonction convexe définie sur un ensemble convexe C. Alors,*
- *tout minimum local de J sur C est un minimum global,*
- *si J est strictement convexe, il y a au plus un minimum global.*

Preuve.
i) Soit $x_0 \in C$ un minimum local, i.e. il existe $r > 0$ tel que $B := B(x_0, r) \subset C$ et

$$\forall x \in B \quad J(x) \geq J(x_0). \tag{9.3}$$

Soit $y \in C$. On considère g définie par

$$g(t) = J(ty + (1-t)x_0). \tag{9.4}$$

Soit $z = \partial B \cap [xy]$. Il existe $\epsilon \in [0,1]$ tel que $z = \epsilon y + (1-\epsilon)x_0$. La relation (9.3) implique que

$$\forall t \in [0, \epsilon] \quad g(t) \geq J(x_0).$$

En utilisant cette dernière inégalité et le caractère convexe de J nous obtenons :

$$\forall t \in [0, \epsilon] \quad J(x_0) \leq g(t) \leq tJ(y) + (1-t)J(x_0),$$

d'où : $t(J(y) - J(x_0)) \geq 0 \quad \forall t \in [0, \epsilon]$, et en prenant $t = \epsilon$ on obtient $J(y) \geq J(x_0)$.

ii) Soit J strictement convexe. On suppose que J admet deux minimums globaux distincts x, y. Soit alors $t \in]0,1[$ et $z = tx + (1-t)y$. On aboutit a la contradiction suivante : $J(z) < tJ(x) + (1-t)J(y) = \min_C J$ □

9.1.4 Conditions d'optimalité

Nous supposons dans tout ce paragraphe que J est un ou deux fois différentiable. On notera $\mathbf{x}^*$ un minimum (local) de J.
Ce qui suit reste valable dans le cas où le minimum $\mathbf{x}^*$ se trouve à l'intérieur de l'ensemble des contraintes. Nous donnons les conditions nécessaires de minimum, puis celles suffisantes.

Théorème 9.2 (Conditions nécessaires) *Les deux conditions nécessaires sont les suivantes*
- *Condition au premier ordre : si J est différentiable en $\mathbf{x}^*$, on a $\nabla J(\mathbf{x}^*) = \mathbf{0}$,*
- *Condition au second ordre : si J est deux fois différentiable au point $\mathbf{x}^*$, alors la forme quadratique $D^2J(\mathbf{x}^*)$ est positive i.e.*

$$\left\langle D^2J(\mathbf{x}^*)\mathbf{y}, \mathbf{y} \right\rangle \geq 0,$$

où $D^2J(\mathbf{x}^*)$ est la matrice hessienne, définie par les coefficients $\frac{\partial^2 J}{\partial_{x_i}\partial_{x_j}}(\mathbf{x}^*)$.

Preuve.
i) on considère un développement limité à l'ordre un en x^* :

$$J(x^* + h) = J(x^*) + \nabla J(x^*)h + \|h\|E(x^*; h).$$

Soit $t \geq 0$. On choisit $h = -t\nabla J(x^*)$ et on pose $\alpha = \|\nabla J(x^*)\|$. Il vient

$$J(x^* + h) = J(x^*) - t\alpha^2 + t\alpha E(x^*; h).$$

Supposons que $\alpha > 0$. Puisque $E(x^*; h) \underset{t\to 0}{\to} 0$ on vérifie alors qu'il existe $t^* > 0$ tel que $\forall t \leq t^* : \quad J(x^* + h) < J(x^*)$. Ceci indique que x^* n'est pas un minimum. Ainsi il est nécéssaire que $\alpha = 0$.

ii) Supposons qu'il existe $y \in \mathbb{R}^n, y \neq 0$ tel que $a := \langle D^2J(\mathbf{x}^*)\mathbf{y}, \mathbf{y}\rangle < 0$. On considère un développement limité à l'ordre deux en x^* :

$$J(x^* + ty) = J(x^*)\frac{1}{2}t^2a + t^2a^2\alpha E_2(x^*; ty).$$

Puisque $E_2(x^*; ty) \underset{t\to 0}{\to} 0$, on vérifie qu'il existe $t^* > 0$ tel que $\forall t \in [0, t^*] : \quad J(x^* + ty) < J(x^*)$. Ceci indique que x^* n'est pas un minimum. Ainsi il est nécéssaire que $D^2J(\mathbf{x}^*)$ soit positive. □

Nous énonçons à présent des conditions nécéssaires et suffisantes pour qu'un point x^* soit un minimum, dans le cas où J est suffisamment régulière.

Théorème 9.3 *Conditions suffisantes.*
Soit J une fonction de classe $\mathcal{C}^1$ définie sur $\mathbb{R}^n$. On suppose que : $\nabla J(\mathbf{x}^*) = \mathbf{0}$ et que J est deux fois différentiable en x^*.
Alors, $\mathbf{x}^*$ est un minimum (local) de J si l'une des deux conditions suivantes est vérifiée
i) $D^2J(\mathbf{x}^*)$ est définie positive,
ii) $\exists r > 0$ tel que J est deux fois différentiable sur $B(x^*, r)$ et, la forme quadratique $D^2J(\mathbf{x})$ est positive pour tout $x \in B(x^*, r)$

Dans le cas où J est convexe, la condition suffisante s'exprime beaucoup plus facilement. En effet, nous avons la

Proposition 9.3 *Soit J une fonction convexe de classe $\mathcal{C}^1$, définie sur $\mathbb{R}^n$ et $\mathbf{x}^*$ un point de $\mathbb{R}^n$. Alors, $\mathbf{x}^*$ est un minimum (global) de J si et seulement si $\nabla J(\mathbf{x}^*) = \mathbf{0}$.*

Preuve.
Il suffit de montrer que la condition est suffisante. Soit y quelconque. D'après la proposition 9.1 nous avons :

$$J(y) - J(x^*) \geq \nabla J(x^*)(y - x^*) = 0,$$

ce qui montre bien que x^* est un minimum. □

9.1.4.1 Contraintes égalités et inégalités

La plupart des problèmes d'optimisation faisant intervenir des contraintes, il est bon de connaitre aussi les conditions d'optimalité dans ce cas. Nous allons donner le principal résultat (dû à Lagrange pour des contraintes égalités et à Kuhn et Tucker dans le cas général) sans démonstration en renvoyant au cours (électif) d'optimisation de deuxième année pour plus de détails. On suppose, dans cette partie, que l'ensemble C sur lequel on veut minimiser J est donné par des contraintes de type égalités et inégalités :

$$C = \{X \in \mathbb{R}^N,\ f_i(X) = 0,\ i \in \{1, 2, \ldots, p\},\ g_i(X) \leq 0,\ i \in \{1, 2, \ldots, m\}\}$$

où les f_j et les g_i sont des fonctions de classe C^1 de $\mathbb{R}^N$ dans $\mathbb{R}$.

On est conduit à introduire la définition suivante :

Définition 9.2 *On dit que les contraintes sont* **qualifiées** *au point X^* si*
- *ou bien les fonctions g_i sont affines*
- *ou bien les vecteurs $\nabla g_i(X^*)$, pour $i \in I_0$, sont linéairement indépendants, où I_0 est l'ensemble des contraintes saturées au point X^* :*

$$I_0 = \{i \in I;\ g_i(X^*) = 0\}.$$

On a alors le résultat fondamental suivant :

Théorème 9.4 (Kuhn-Tucker, Lagrange) *Soit J, f_i, $i \in \{1, 2, \ldots, p\}$, g_i, $i \in \{1, 2, \ldots, m\}$ des fonctions de classe C^1. On veut minimiser J sur l'ensemble $C = \{X \in \mathbb{R}^N,\ f_i(X) = 0,\ i \in \{1, 2, \ldots, p\},\ g_i(X) \leq 0,\ i \in \{1, 2, \ldots, m\}\}$. On suppose les contraintes qualifiées au point X^*. Alors une* **condition nécessaire** *pour que X^* soit un minimum de J est qu'il existe des nombres positifs $\lambda_1, \lambda_2, \ldots, \lambda_m$ et des nombres réels $\mu_1, \mu_2, \ldots, \mu_p$ tels que*

$$\begin{cases} \nabla J(X^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(X^*) = \sum_{i=1}^{p} \mu_i \nabla f_i(X^*) \\ \text{avec } \lambda_i g_i(X^*) = 0 \text{ pour } i \in \{1, 2, \ldots, m\}. \end{cases} \tag{9.5}$$

Remarque 9.2 *Les nombres (inconnus) λ_i et μ_i sont appelés des multiplicateurs. Dans la pratique, on est donc conduit à résoudre un système* **non linéaire** *de $N + p + m$ équations à $N + p + m$ inconnues (les N coordonnées de X^*, les p multiplicateurs μ_i et les m multiplicateurs λ_i) :*
- *N équations viennent de l'égalité $\nabla J(X^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(X^*) = \sum_{i=1}^{p} \mu_i \nabla f_i(X^*)$,*
- *p équations viennent de $f_i(X) = 0$, $i = 1, \ldots p$,*
- *m équations viennent de $\lambda_i g_i(X^*) = 0$, $i = 1, \ldots m$.*

Une technique possible de résolution du problème d'optimisation est effectivement de résoudre ce système non linéaire par l'une des méthodes vues au chapitre précédent. Dans la pratique c'est peu utilisé, car c'est assez lourd et ce système a bien souvent beaucoup d'autres solutions (tous les points critiques) que celle qui nous intéresse.

9.2 Algorithmes d'optimisation mono-dimensionnels ou recherche du pas

Une grande classe d'algorithmes que nous allons considérer pour les problèmes d'optimisation ont la forme générale suivante

$$\mathbf{x}^{(0)} \text{ étant donné, calculer } \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{9.6}$$

Le vecteur $\mathbf{d}^{(k)}$ s'appelle la direction de descente, $\rho^{(k)}$ le pas de la méthode à la k-ième itération. En pratique, on s'arrange presque toujours pour satisfaire l'inégalité

$$J(\mathbf{x}^{(k+1)}) \leq J(\mathbf{x}^{(k)}).$$

De tels algorithmes sont souvent appelés méthodes de descente. Essentiellement, la différence entre ces algorithmes réside dans le choix de la direction de descente $\mathbf{d}^{(k)}$. Cette direction étant choisie, nous sommes plus ou moins ramené à un problème unidimensionnel pour la détermination de $\rho^{(k)}$. Pour ces raisons, commençons par analyser ce qui se passe dans le cas de la dimension un.
Soit $q(\rho)$ la fonction coût que l'on cherche à minimiser. On pourra prendre par exemple

$$q(\rho) = J(\mathbf{x}^{(k)} + \rho\mathbf{d}^{(k)})$$

afin d'appliquer les idées au cas de la méthode de descente. Supposons que l'on connaisse un intervalle $[a; b]$ contenant le minimum ρ^* de q et tel que q soit décroissante sur $[a; \rho^*]$ et croissante sur $]\rho^*; b]$ (q est alors appelée une fonction unimodale).

9.2.1 Méthode de la section dorée

On construit une suite décroissante d'intervalles $[a_i; b_i]$ qui contiennent tous le minimum ρ^*. Pour passer de $[a_i; b_i]$ à $[a_{i+1}; b_{i+1}]$, on procède de la manière suivante. On introduit deux nombres a' et b' de l'intervalle $[a_i; b_i]$ et tels que $a' < b'$. Puis, on calcule les valeurs $q(a')$ et $q(b')$. Trois possibilités se présentent alors à nous. Si $q(a') < q(b')$, alors, le minimum ρ^* se trouve nécessairement à gauche de b'. Ceci définit alors le nouvel intervalle en posant $a_{i+1} = a_i$ et $b_{i+1} = b'$. Considérons maintenant que l'inégalité : $q(a') > q(b')$ est satisfaite. Dans ce second cas, il est évident que le minimum se trouve cette fois à droite de a'. On pose alors : $a_{i+1} = a'$ et $b_{i+1} = b_i$. Enfin, le dernier cas consiste à avoir $q(a') = q(b')$. Alors, le minimum se trouve dans l'intervalle $[a'; b']$. On se restreint donc à $a_{i+1} = a'$ et $b_{i+1} = b'$.
La question suivante se pose : comment choisir a' et b' en pratique ? En général, on privilégie deux aspects :

i) on souhaite que le facteur de réduction τ, qui représente le ratio du nouvel intervalle par rapport au précédent, soit constant,
ii) on désire réutiliser le point qui n'a pas été choisi dans l'itération précédente afin de diminuer les coûts de calculs.

On peut montrer que la vérification simultanée de ces deux contraintes conduit à un choix unique des paramètres a' et b'. Plus précisément, supposons que q est unimodale. Alors, on obtient l'algorithme de la table 9.1 dit de la section dorée, la méthode tirant son nom de la valeur du paramètre τ.
Ici, $N^{\max}$ est le nombre maximal d'itérations que l'on se fixe. A cette fin, on doit valider un critère d'arrêt de la forme : $|b_{i+1} - a_{i+1}| < \varepsilon$, où ε est l'erreur (ou tolérance) que l'on se permet sur la solution ρ^* du problème.

poser $\tau = \frac{1+\sqrt{5}}{2}$
poser $a_0 = a$
poser $b_0 = b$
pour $i = 0, ..., N^{\max}$
    poser $a' = a_i + \frac{1}{\tau^2}(b_i - a_i)$
    poser $b' = a_i + \frac{1}{\tau}(b_i - a_i)$
    si $(q(a') < q(b'))$ alors
        poser $a_{i+1} = a_i$
        poser $b_{i+1} = b'$
    sinon si $(q(a') > q(b'))$ alors
        poser $a_{i+1} = a'$
        poser $b_{i+1} = b_i$
    sinon si $(q(a') = q(b'))$ alors
        poser $a_{i+1} = a'$
        poser $b_{i+1} = b'$
    fin si
fin pour i

TABLE 9.1 – Algorithme de la section dorée.

9.2.2 Méthode d'interpolation parabolique

L'idée maîtresse de la méthode d'interpolation parabolique consiste à remplacer la fonction coût q par son polynôme d"interpolation p d'ordre deux (d'où l'appellation d'interpolation parabolique) en trois points x_0, y_0 et z_0 de l'intervalle $[a; b]$. Ces points sont choisis tels que : $q(x_0) \geq q(y_0)$ et $q(z_0) \geq q(y_0)$. On peut montrer que si nous posons

$$q[x_0; y_0] = \frac{q(y_0) - q(x_0)}{y_0 - x_0},$$

et

$$q[x_0; y_0; z_0] = \frac{q[z_0; y_0] - q[x_0; y_0]}{z_0 - x_0},$$

alors, le minimum est donné par

$$y_1 = \frac{x_0 + y_0}{2} - \frac{q[x_0; y_0]}{2q[x_0; y_0; z_0]}.$$

Il est clair que $\rho^* \in [x_0; z_0]$ selon les choix précédents. On choisit ensuite les trois nouveaux points de la manière suivante
- si $y_1 \in [x_0; y_0]$, on pose alors $x_1 = x_0$, $y_1 = y_1$ et $z_1 = y_0$ puisque $\rho^* \in [x_0; y_0]$,
- si $y_1 \in [y_0; z_0]$, on pose alors $x_1 = y_0$, $y_1 = y_1$ et $z_1 = z_0$ car $\rho^* \in [y_0; z_0]$.

Puis on recommence. Ceci conduit à l'algorithme donné table 9.2.
On peut montrer que la méthode est d'ordre 1.3. En fait, nous pouvons montrer qu'il existe une constante strictement positive C, indépendante de i, telle que l'on ait l'inégalité

$$|y_{i+1} - \rho^*| \leq C|y_i - \rho^*|^{1.3}.$$

choisir x_0, y_0 et z_0 dans $[a;b]$ tels que $q(x_0) \geq q(y_0)$ et $q(z_0) \geq q(y_0)$
pour $i = 0, ..., N^{\max}$
    poser $q[x_i; y_i] = \frac{q(y_i)-q(x_i)}{y_i-x_i}$
    poser $q[x_i; y_i; z_i] = \frac{q[x_i;z_i]-q[x_i;y_i]}{z_i-x_i}$
    poser $y_{i+1} = \frac{x_i+y_i}{2} - \frac{q[x_i;y_i]}{2q[x_i;y_i;z_i]}$
    si $y_{i+1} \in [x_i; y_i]$ alors
        poser $x_{i+1} = x_i$
        poser $z_{i+1} = y_i$
    sinon si $y_{i+1} \in [y_i; z_i]$ alors
        poser $x_{i+1} = y_i$
        poser $z_{i+1} = z_i$
    fin si
fin pour i

TABLE 9.2 – Algorithme de l'interpolation parabolique.

Dire que la méthode est d'ordre 1.3 signifie que si à une étape donnée l'erreur de 10^{-2}, elle sera de l'ordre de $(10^{-2})^{1.3} \approx 2.5\ 10^{-3}$ à l'étape suivante.
Une des difficultés concerne l'initialisation de l'algorithme. Pratiquement, on peut procéder de la façon suivante. On choisit un point α_0 de l'intervalle $[a;b]$ ainsi qu'un pas de déplacement positif δ. On calcule ensuite $q(\alpha_0)$ et $q(\alpha_0 + \delta)$. On a alors deux situations

- si $q(\alpha_0) \geq q(\alpha_0 + \delta)$, alors q décroît et donc ρ^* est à droite de α_0. On continue alors à calculer $q(\alpha_0 + 2\delta)$, $q(\alpha_0 + 3\delta)$,...,$q(\alpha_0 + k\delta)$ jusqu'à tomber sur un entier k tel que q croît : $q(\alpha_0 + k\delta) > q(\alpha_0 + (k-1)\delta)$, avec $k \geq 2$. On pose alors

$$x_0 = \alpha_0 + (k-2)\delta, y_0 = \alpha_0 + (k-1)\delta, z_0 = \alpha_0 + k\delta.$$

- si $q(\alpha_0) < q(\alpha_0 + \delta)$, alors ρ^* est à gauche de $\alpha_0 + \delta$. On prend $-\delta$ comme pas jusqu'à tomber sur un entier k tel que : $q(\alpha_0 - k\delta) \geq q(\alpha_0 - (k-1)\delta)$. On pose alors

$$x_0 = \alpha_0 - k\delta, y_0 = \alpha_0 - (k-1)\delta, z_0 = \alpha_0 - (k-2)\delta.$$

9.2.3 D'autres règles

La recherche du pas n'est qu'une étape d'un algorithme plus complexe pour minimiser J. La philosophie générale est alors de plutôt essayer d'avoir une approximation satisfaisante du pas optimal. Si nous considérons $q(\rho) = J(\mathbf{x} + \rho\mathbf{d})$, avec $\rho > 0$, nous avons, par application de la règle de dérivation en chaîne

$$q'(\rho) = \nabla J(\mathbf{x} + \rho\mathbf{d}) \cdot \mathbf{d}.$$

Par conséquent, puisque $\mathbf{d}$ est une direction de descente, $q'(0) = \nabla J(\mathbf{x}) \cdot \mathbf{d} < 0$. Il s'ensuit que si nous prenons ρ un peu à droite de 0, on est sûr de faire décroître q. Toutefois, il faut faire attention car deux éléments contradictoires sont à prendre en compte

- si ρ est trop grand, on risque de ne pas faire décroître la fonction q ou son comportement peut être oscillant,

• si ρ est trop petit, l'algorithme n'avancera pas assez vite.
Il existe deux règles classiques pour parvenir à cette fin, la règle de Goldstein (1967) et celle de Wolfe (1969).

9.3 Quelques notions sur les algorithmes

Intéressons nous maintenant au développement d'algorithmes numériques de résolution des problèmes de minimisation destinés à être mis en oeuvre sur calculateurs. Nous ne verrons ici que des algorithmes de base, l'optimisation étant un vaste domaine de recherches et d'applications. Nous ne nous intéresserons ici qu'à l'optimisation locale, la recherche d'un extremum global étant hors de portée de cette introduction. Par ailleurs, l'hypothèse de différentiabilité nous suivra tout le long de cet exposé ; encore une fois, l'optimisation non différentiable n'est pas traitée ici.
Rappelons ce que l'on entend par algorithme.

Définition 9.3 *Un algorithme itératif est défini par une application vectorielle* $\mathbb{A} : \mathbb{R}^n \to \mathbb{R}^n$ *qui génère une suite de vecteurs* $(\mathbf{x}^{(k)})_{k\geq 0}$ *par une construction typiquement de la forme*

`choisir` $\mathbf{x}^{(0)}$ `(phase d'initialisation de l'algorithme)`
`calculer` $\mathbf{x}^{(k+1)} = \mathbb{A}(\mathbf{x}^{(k)})$ `(`k`-ième itération))`

Bien sûr, ce que nous espérons, c'est que notre suite $(\mathbf{x}^{(k)})_{k\geq 0}$ converge vers une limite $\mathbf{x}^*$ qui sera effectivement notre point de minimum relatif. On dit que l'algorithme converge vers la solution du problème de minimisation si c'est le cas. Lorsque l'on a un algorithme donné, deux mesures importantes de son efficacité sont : d'une part la vitesse de convergence, d'autre part, sa complexité calculatoire. La vitesse de convergence mesure "la rapidité" avec laquelle la suite $(\mathbf{x}^{(k)})_{k\geq 0}$ converge vers le point $\mathbf{x}^*$. La complexité mesure le coût des opérations nécessaires pour obtenir une itération, le coût global étant le coût d'une itération multiplié par le nombre d'itérations pour obtenir la solution escomptée avec une certaine précision ε fixée *a priori*. On prend généralement les appellations suivantes, semblables à celles du chapitre précédent. On introduit l'erreur vectorielle sur la solution : $\mathbf{e}^{(k)} = \mathbf{x}^* - \mathbf{x}^{(k)}$. Si sa norme (euclidienne) $e^{(k)} = \left\|\mathbf{e}^{(k)}\right\|$ décroît linéairement, alors, on dit que la vitesse de convergence est linéaire. Plus mathématiquement, cette propriété s'exprime par une relation du type

$$(\exists C \in [0;1[)(\exists k_0 \in \mathbb{N})(\forall k \geq k_0)(e^{(k+1)} \leq Ce^{(k)})$$

On voit bien à ce niveau que la vitesse de convergence est un notion asymptotique. Elle n'est pas nécessairement observable à la première itération. Si nous observons une relation du type $e^{(k+1)} \leq \gamma^{(k)} e^{(k)}$, nous dirons que la vitesse est superlinéaire si $\lim_{k\to+\infty} \gamma^{(k)} = 0$, pour $\gamma^{(k)} \geq 0$, pour tout $k \geq 0$. On parle de convergence géométrique lorsque la suite $(\gamma^{(k)})_k$ est une suite géométrique. La méthode est dite d'ordre p si l'on a une relation du type

$$(\exists C \in [0;1[)(\exists k_0 \in \mathbb{N})(\forall k \geq k_0)(e^{(k+1)} \leq C(e^{(k)})^p)$$

Si $p = 2$, nous dirons que la vitesse de convergence est quadratique. Finalement, si la convergence a lieu seulement pour des $\mathbf{x}^{(0)}$ voisins de $\mathbf{x}^*$, nous parlerons de convergence locale, sinon, nous dirons globale.

```
poser k = 0
choisir x^(0)
tant que (‖x^(k+1) − x^(k)‖ ≥ ε) et (k ≤ k^max) faire
      calculer d^(k) = −∇J(x^(k))
      calculer ρ^(k)
      poser x^(k+1) = x^(k) + ρ^(k) d^(k)
fin tant que
```

TABLE 9.3 – Algorithme du gradient.

9.4 Méthodes de gradient

La méthode du gradient fait partie des classes de méthodes dites de descente. Quelle est l'idée cachée derrière ces méthodes ? Considérons un point de départ $\mathbf{x}^{(0)}$ et cherchons à minimiser une fonction J. Puisque l'on veut atteindre $\mathbf{x}^*$, nous cherchons à avoir : $J(\mathbf{x}^{(1)}) < J(\mathbf{x}^{(0)})$. Une forme particulièrement simple est de chercher $\mathbf{x}^{(1)}$ tel que le vecteur $\mathbf{x}^{(1)} - \mathbf{x}^{(0)}$ soit colinéaire à une direction de descente $\mathbf{d}^{(0)} \neq \mathbf{0}$. Nous le noterons : $\mathbf{x}^{(1)} - \mathbf{x}^{(0)} = \rho^{(0)}\mathbf{d}^{(1)}$, où $\rho^{(0)}$ est le pas de descente de la méthode. On peut alors itérer de cette manière en se donnant $\mathbf{x}^{(k)}$, $\mathbf{d}^{(k)}$ et $\rho^{(k)}$ pour atteindre $\mathbf{x}^{(k+1)}$ par

```
choisir x^(0)
calculer x^(k+1) = x^(k) + ρ^(k) d^(k)
```

avec $\mathbf{d}^{(k)} \in \mathbb{R}^{n*}$ et $\rho^{(k)} > 0$. De nombreux choix existent pour $\mathbf{d}^{(k)}$ et $\rho^{(k)}$.
La première question consiste à choisir la direction de descente. Rappelons que le développement de Taylor de J au premier ordre donne au voisinage de $\mathbf{x}^{(k+1)}$

$$J(\mathbf{x}^{(k+1)}) = J(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = J(\mathbf{x}^{(k)}) + \rho^{(k)}\nabla J(\mathbf{x}^{(k)}) \cdot \mathbf{d}^{(k)} + \rho^{(k)})\|\mathbf{d}^{(k)}\| E(\mathbf{x}^{(k)}; \rho^{(k)}\mathbf{d}^{(k)}),$$

où $\lim_{\rho^{(k)}\mathbf{d}^{(k)} \to 0} E(\mathbf{x}^{(k)}; \rho^{(k)}\mathbf{d}^{(k)}) = 0$. Or, puisque l'on désire avoir : $J(\mathbf{x}^{(1)}) < J(\mathbf{x}^{(0)})$, une solution évidente consiste à prendre

$$\mathbf{d}^{(k)} = -\nabla J(\mathbf{x}^{(k)}),$$

puisqu'alors

$$J(\mathbf{x}^{(k+1)}) - J(\mathbf{x}^{(k)}) = -\rho^{(k)} \left\| \nabla J(\mathbf{x}^{(k)}) \right\|^2 + o(\rho^{(k)}).$$

Nous voyons que si $\rho^{(k)}$ est suffisamment petit, $\mathbf{x}^{(k+1)}$ minimisera mieux J que ne le faisait $\mathbf{x}^{(k)}$. La méthode obtenue avec le choix $\mathbf{d}^{(k)} = -\nabla J(\mathbf{x}^{(k)})$ est appelée méthode du gradient. Lorsque l'on travaille sur une résolution numérique d'un problème, on se donne en général un critère d'arrêt de la forme : $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \varepsilon$. De plus, puisque la convergence n'est pas toujours assurée, une règle de base est de fixer un nombre maximum d'itérations $k^{\max}$. On obtient alors l'algorithme présenté table 9.3 et dit du gradient.
Même si ces méthodes sont conceptuellement très simples et qu'elles peuvent être programmées directement, elles sont souvent lentes dans la pratique. Elles convergent mais sous des conditions de convergence souvent complexes. A titre d'exemple, donnons le résultat suivant.

Théorème 9.5 *Soit J une fonction de classe $\mathcal{C}^1$ de $\mathbb{R}^n$ dans $\mathbb{R}$, $\mathbf{x}^*$ un minimum de J. Supposons que*
i) J est α-elliptique, c'est-à-dire,

$$(\exists \alpha > 0)(\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n)(\nabla J(\mathbf{x}) - \nabla J(\mathbf{y})) \cdot (\mathbf{x} - \mathbf{y}) \geq \alpha \|\mathbf{x} - \mathbf{y}\|^2 .$$

ii) l'application ∇J est lipschitzienne

$$(\exists M > 0)(\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n)(\|\nabla J\mathbf{x} - \nabla J\mathbf{y}\| \leq M \|\mathbf{x} - \mathbf{y}\|).$$

S'il existe deux réels a et b tels que $\rho^{(k)}$ satisfasse $0 < a < \rho^{(k)} < b < \frac{2\alpha}{M^2}$, pour tout $k \geq 0$, alors, la méthode du gradient définie par

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \rho^{(k)} \nabla J(\mathbf{x}^{(k)})$$

converge pour tout choix de $\mathbf{x}^{(0)}$ de façon linéaire (ou géométrique)

$$\exists \beta \in]0; 1[, \left\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \right\| \leq \beta^k \left\| \mathbf{x}^{(0)} - \mathbf{x}^* \right\| .$$

Le choix du pas $\rho^{(k)}$ peut être effectué de la manière suivante
- soit $\rho^{(k)} = \rho$ est fixé *a priori* : c'est ce que l'on appelle la méthode du gradient à pas fixe ou constant,
- soit $\rho^{(k)}$ est choisi comme le minimum de la fonction $q(\rho) = J(\mathbf{x}^{(k)} - \rho \nabla J(\mathbf{x}^{(k)}))$: c'est ce que l'on appelle la méthode du gradient à pas optimal,
- ou, soit $\rho^{(k)}$ est calculé par les méthodes de recherche du pas présentées précédemment.

Dans le cas du gradient à pas optimal, nous avons le même résultat de convergence que précédemment sous des hypothèses plus faibles sur J.

Théorème 9.6 *Soit J une fonction de classe $\mathcal{C}^1$ de $\mathbb{R}^n$ dans $\mathbb{R}$, $\mathbf{x}^*$ un minimum de J. On suppose que J est α-elliptique. Alors, la méthode du gradient à pas optimal converge pour tout choix du vecteur d'initialisation $\mathbf{x}^{(0)}$.*

Remarque 9.3 *Même pour le gradient à pas optimal qui est en principe la meilleure de ces méthodes d'un point de vue de la rapidité de convergence, celle-ci peut être lente car altérée par un mauvais conditionnement de la matrice hessienne de J. Par ailleurs, on peut considérer des critères de convergence sur le gradient de J en $\mathbf{x}^{(k)}$: $\left\|\nabla J(\mathbf{x}^{(k)})\right\| < \varepsilon_1$*

9.5 Méthode du gradient conjugué

Considérons une matrice A, définie positive, et soit J la fonctionnelle quadratique définie par

$$\begin{aligned} J : \mathbb{R}^n &\to \mathbb{R} \\ \mathbf{x} &\mapsto J(\mathbf{x}) = \frac{1}{2} A\mathbf{x} \cdot \mathbf{x} - \mathbf{b} \cdot \mathbf{x}. \end{aligned}$$

La fonction J est alors une fonctionnelle strictement convexe (à montrer) deux fois continûment différentiable. Un calcul de son gradient donne : $\nabla J(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$. Par conséquent, le minimum (unique et global) de J est réalisé en $\mathbf{x}^*$ tel que : $A\mathbf{x}^* = \mathbf{b}$. La méthode que nous allons voir ici s'avère d'une redoutable efficacité pour résoudre le système linéaire $A\mathbf{x}^* = \mathbf{b}$.

Afin de développer le gradient conjugué, introduisons la notion de direction conjuguée.

Définition 9.4 *Nous dirons que deux vecteurs (ou directions)* $\mathbf{d}_1$ *et* $\mathbf{d}_2$ *sont conjugués pour la matrice* A *si :* $A\mathbf{d}_2 \cdot \mathbf{d}_1 = 0$.

Ceci signifie que ces deux vecteurs sont orthogonaux pour le produit scalaire associé à la matrice A, défini par

$$(\mathbf{x}, \mathbf{y})_A = A\mathbf{x} \cdot \mathbf{y}, \quad \forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n.$$

Faisons l'hypothèse que nous connaissons k directions conjuguées $\mathbf{d}^{(0)}, ..., \mathbf{d}^{(k-1)}$. La méthode de descente consiste, en partant d'un point $\mathbf{x}^{(0)} \in \mathbb{R}^n$, à calculer par des itérations successives $\mathbf{x}^{(k+1)}$ tel qu'il satisfasse

$$J(\mathbf{x}^{(k+1)}) = J(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \min_{\rho \in \mathbb{R}} J(\mathbf{x}^{(k)} + \rho \mathbf{d}^{(k)}).$$

On peut expliciter la valeur de $\rho^{(k)}$ en utilisant la condition de minimum au premier ordre, à savoir

$$\nabla J(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) \cdot \mathbf{d}^{(k)} = 0,$$

et plus explicitement dans notre cas

$$\rho^{(k)} = -\frac{(A\mathbf{x}^{(k)} - \mathbf{b}) \cdot \mathbf{d}^{(k)}}{\left\|\mathbf{d}^{(k)}\right\|_A^2} = -\frac{(\mathbf{x}^{(k)}, \mathbf{d}^{(k)})_A}{\left\|\mathbf{d}^{(k)}\right\|_A^2} + \frac{\mathbf{b} \cdot \mathbf{d}^{(k)}}{\left\|\mathbf{d}^{(k)}\right\|_A^2} \tag{9.7}$$

en posant $\left\|\mathbf{d}^{(k)}\right\|_A = (\mathbf{d}^{(k)}, \mathbf{d}^{(k)})_A^{1/2}$. Or, par définition, $\mathbf{x}^{(k)} - \mathbf{x}^{(0)}$ peut s'exprimer selon les vecteurs $\mathbf{d}^{(0)}, ..., \mathbf{d}^{(k-1)}$ puisque

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \rho^{(k-1)}\mathbf{d}^{(k-1)} = \mathbf{x}^{(k-2)} + \rho^{(k-2)}\mathbf{d}^{(k-2)} + \rho^{(k-1)}\mathbf{d}^{(k-1)} = ... = \mathbf{x}^{(0)} + \sum_{\ell=0}^{k-1} \rho^{(\ell)}\mathbf{d}^{(\ell)}$$

Ainsi, nous pouvons simplifier (9.7) par conjugaison pour écrire

$$\rho^{(k)} = -\frac{(\mathbf{x}^{(0)}, \mathbf{d}^{(k)})_A}{\left\|\mathbf{d}^{(k)}\right\|_A^2} + \frac{\mathbf{b} \cdot \mathbf{d}^{(k)}}{\left\|\mathbf{d}^{(k)}\right\|_A^2} = -\frac{(\mathbf{r}^{(0)}, \mathbf{d}^{(k)})_A}{\left\|\mathbf{d}^{(k)}\right\|_A^2},$$

où le vecteur résidu $\mathbf{r}^{(0)}$ à l'instant initial est défini par : $\mathbf{r}^{(0)} = A\mathbf{x}^{(0)} - \mathbf{b}$.
Le succès de l'algorithme du gradient conjugué est intimement lié à la proposition importante suivante.

Proposition 9.4 *Le point* $\mathbf{x}^{(k)}$ *est le minimum de* J *sur le sous-espace affine passant par* $\mathbf{x}^{(0)}$ *engendré par les vecteurs* $\{\mathbf{d}^{(0)}, ..., \mathbf{d}^{(k-1)}\}$.

Une conséquence fondamentale de la proposition précédente est que, si l'on est capable de trouver n directions conjuguées $\{\mathbf{d}^{(0)}, ..., \mathbf{d}^{(n-1)}\}$, on a résolu le problème de minimisation puisque $\{\mathbf{d}^{(0)}, ..., \mathbf{d}^{(n-1)}\}$ est une base de $\mathbb{R}^n$. En fait, l'algorithme du gradient conjugué consiste à construire simultanément ces directions conjuguées par le procédé de Gram-Schmidt. Plus précisément, l'algorithme est décrit dans la table 9.4.
L'algorithme de Gram-Schmidt peut, dans certains cas, s'avérer instable. Á cause des erreurs d'arrondis, la méthode peut mettre un peu plus que n itérations pour converger.
Rappelons que l'algorithme présenté ici était particularisé au cas d'une fonctionnelle quadratique. On peut étendre l'algorithme pour une fonctionnelle J quelconque de manière efficace comme nous le détaillons dans la table 9.5.

$k = 0$
choisir $\mathbf{x}^{(0)} \in \mathbb{R}^n$
choisir $\varepsilon > 0$
choisir $\varepsilon_1 > 0$
poser $\mathbf{r}^{(0)} = \nabla J(\mathbf{x}^{(0)})$
tant que $\left(\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \geq \varepsilon\right)$ et $(k \leq k^{\max})$ faire
    si $\left(\left\|\mathbf{r}^{(k)}\right\| < \varepsilon_1\right)$ alors arrêt
    sinon
        si $(k = 0)$ alors
            poser $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$
        sinon
            calculer $\alpha^{(k)} = -\dfrac{(\mathbf{r}^{(k)}, \mathbf{d}^{(k-1)})_A}{\left\|\mathbf{d}^{(k-1)}\right\|_A^2}$
            poser $\mathbf{d}^{(k)} = \mathbf{r}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k-1)}$
        fin si
        calculer $\rho^{(k)} = -\dfrac{(\mathbf{r}^{(k)}, \mathbf{d}^{(k)})}{\left\|\mathbf{d}^{(k)}\right\|_A^2}$
        poser $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
        calculer $\mathbf{r}^{(k+1)} = A\mathbf{x}^{(k+1)} - \mathbf{b}$
        poser $k = k + 1$
    fin si
fin tant que

TABLE 9.4 – Algorithme du gradient conjugué pour une fonctionnelle quadratique.

$k = 0$
choisir $\mathbf{x}^{(0)} \in \mathbb{R}^n$
choisir $\varepsilon > 0$
choisir $\varepsilon_1 > 0$
poser $\mathbf{r}^{(0)} = \nabla J(\mathbf{x}^{(0)})$
tant que $(\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \geq \varepsilon)$ et $(k \leq k^{\max})$ faire
    si $(\left\|\mathbf{r}^{(k)}\right\| < \varepsilon_1)$ alors arrêt
    sinon
        si $(k = 0)$ alors
            poser $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$
        sinon
            calculer $\alpha^{(k)} = \dfrac{\left\|\mathbf{r}^{(k)}\right\|^2}{\left\|\mathbf{r}^{(k-1)}\right\|^2}$
            poser $\mathbf{d}^{(k)} = \mathbf{r}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k-1)}$
        fin si
        si $(\mathbf{d}^{(k)} \cdot \mathbf{r}^{(k)} \geq 0)$ alors
            poser $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$
        sinon
            rechercher un pas $\rho^{(k)}$ approchant le minimum de $J(\mathbf{x}^{(k)} + \rho\mathbf{d}^{(k)})$ par
            $\nabla J(\mathbf{x}^{(k)} + \rho\mathbf{d}^{(k)}) \cdot \mathbf{d}^{(k)} = 0$
        fin si
        poser $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}$
        poser $\mathbf{r}^{(k+1)} = \nabla J(\mathbf{x}^{(k+1)})$
        poser $k = k + 1$
    fin si
fin tant que

TABLE 9.5 – Algorithme du gradient conjugué pour une fonctionnelle générale.

9.6 Approche par la méthode de Newton

La méthode de Newton n'est pas a proprement parler une méthode d'optimisation comme nous l'avons vu dans le chapitre 8. C'est une méthode de recherche de zéros d'une fonction $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ selon : $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. L'idée de cette méthode ici est de résoudre l'équation $\nabla J(\mathbf{x}) = \mathbf{0}$, condition nécessaire de premier ordre pour la détection d'extrema (sans contraintes) d'une fonction. L'équation $\nabla J(\mathbf{x}) = \mathbf{0}$ est donnée par un système $n \times n$ d'équations non linéaires. La méthode s'écrit formellement

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [D^2 J(\mathbf{x}^{(k)})]^{-1} \nabla J(\mathbf{x}^{(k)}).$$

Un des gros problèmes de cette méthode est que *le choix de* $\mathbf{x}^{(0)}$ *joue un grand rôle sur la convergence ou non de la méthode de Newton*. La méthode est très sensible à l'initialisation. Il peut aussi arriver que la méthode converge vers un extremum qui n'est pas celui cherché. Une approche possible consiste à faire tourner quelques itérations d'une méthode de gradient pour approcher $\mathbf{x}^*$ et de considérer l'itéré résultant comme le point de départ de la méthode de Newton. L'avantage de la méthode de Newton est sa grande rapidité de convergence : *la convergence de la méthode de Newton est quadratique*. Un autre inconvénient de la méthode de Newton est son coût en terme de calcul (évaluation des N^2 termes de la matrice Hessienne) qui fait qu'on lui préfère bien souvent des méthodes de type quasi-Newton où cette matrice Hessienne est approchée, dans l'esprit de la section 8.3.2. Dans cet esprit les méthodes dites BFGS sont parmi les plus utilisées.

9.7 Exercices du chapitre 9

Exercice 9.1 Dans le Théorème 9.1, montrer par des contre-exemples, que les hypothèses J continue ou C fermé sont indispensables.

Exercice 9.2 Soit N une norme sur un espace vectoriel E. Montrer que N (et plus généralement N^α où $\alpha \geq 1$) est une fonction convexe. Est-elle toujours strictement convexe ? Donnez des exemples où elle l'est et des exemples où elle ne l'est pas.

Exercice 9.3 Soit A une matrice de dimension $n \times m$, b un vecteur fixé de $\mathbb{R}^n$ et F la fonction définie sur $\mathbb{R}^m$ par

$$F(X) = \|AX - b\|_2^2$$

où $\|.\|_2$ est la norme euclidienne classique. Montrer que F est convexe, à quelle condition possède-t-elle un minimum sur $\mathbb{R}^m$? Calculer le gradient de F et sa matrice hessienne, puis déterminer le minimum dans ce cas.

Exercice 9.4 Rechercher le rectangle de périmètre fixé P qui a l'aire maximale. Même question pour un triangle.
Rechercher le parallélépipède de surface fixée S qui a le volume maximal. Même question pour un tétraèdre.

Exercice 9.5 Soit la fonction $F(x, y) = \frac{5}{2}x^2 + y^2 - 4xy$ et l'ensemble $E = \{(x, y) \in \mathbb{R}^2, y^2 \leq x, x \leq 1\}$.
1) Dessiner l'ensemble E et montrer que F possède un minimum sur E.
2) Déterminer ce minimum en utilisant les conditions de Kuhn-Tucker.

9.8 TD9 : optimisation

Exercice 1 Soit la fonction $F(x,y) = 3xy^2 - 2x^3$ et l'ensemble $E = \{(x,y) \in \mathbb{R}^2, x^2 \leq y, y \leq x\}$.
Montrer que F possède un minimum sur E et le déterminer en utilisant les conditions d'optimalité de Kuhn-Tucker.

Exercice 2
Soit $n \in \mathbb{N}^*$. On considère la matrice $A_n \in \mathcal{M}_n(\mathbb{R})$ et le vecteur $b_n \in \mathbb{R}^n$ définis par :

$$A_n = \begin{pmatrix} 4 & -2 & 0 & \dots & 0 \\ -2 & 4 & -2 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -2 & 4 & -2 \\ 0 & \dots & 0 & -2 & 4 \end{pmatrix} \text{ et } b_n = (1, 1, \dots, 1).$$

On cherche à minimiser dans $\mathbb{R}^n$, par différentes méthodes, la fonctionnelle :

$$J_n(x) = \frac{1}{2}(A_n x, x) - (b_n, x).$$

On appelle donc $(\mathcal{P}_n)$ le problème :

$$(\mathcal{P}_n) \begin{cases} \min J_n(x) \\ x \in \mathbb{R}^n \end{cases}$$

Remarque : il est important d'exploiter, dans les questions qui suivent, le format creux de la matrice A afin de diminuer les temps de calcul.
Fonctions Matlab utiles : *spdiags*.

1. On considère le cas $n = 2$. Montrer que J_2 est coercive sur $\mathbb{R}^2$ et résoudre le problème $(\mathcal{P}_2)$.
2. Nous allons étudier deux méthodes de minimisation. Pour chacune de ces études, on demande :

   $\longrightarrow$ lorsque n prend les valeurs 10, 20, 30, 50, 100 :
   - de tester chacune des trois méthodes
   - enfin, de comparer à l'aide d'un graphique ou d'un tableau, la rapidité de convergence de chacune de ces méthodes, ainsi que le temps de calcul par Matlab, suivant les différentes valeurs prises par n.

   (a) **La méthode du gradient à pas fixe.**
   Écrire une fonction Matlab prenant en argument $\rho > 0$, un pas fixe et $x^0 \in \mathbb{R}^n$, un vecteur d'initialisation, afin de mettre en œuvre l'algorithme du gradient à pas fixe, puis la tester sur J_n dans chacun des cas ci-dessus. Répondre aux questions initiales. Expliquer brièvement pourquoi il est important de choisir le pas fixe, ni trop grand, ni trop petit.

(b) **La méthode du gradient conjugué dans le cas d'une fonctionnelle quadratique elliptique.**
Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}$
$x \longmapsto \frac{1}{2}(Ax, x) - (b, x)$
où $A \in \mathcal{M}_n(\mathbb{R})$ est une matrice symétrique définie positive et b est un vecteur de $\mathbb{R}^n$. Alors, dans ce cas, l'algorithme du gradient conjugué s'écrit :

$$\begin{cases} x^0 \text{ est donné, } r^0 = Ax^0 - b \text{ et } d^0 = -r^0. \\ x^{n+1} = x^n + \rho_n d^n, \ \rho_n = -\dfrac{(r^n, d^n)}{(Ad^n, d^n)} \\ r^{n+1} = Ax^{n+1} - b \\ d^{n+1} = -r^{n+1} + \beta_n d^n, \ \beta_n := \dfrac{\|r^{n+1}\|^2}{\|r^n\|^2}. \end{cases}$$

Programmer cet algorithme et répondre aux questions initiales.

9.9 Aide-mémoire

9.9.1 Topologie

Dans cette section, je (re)donne les définitions des objets topologiques qui sont manipulés dans ce cours. Quand je l'estime utile, je donne aussi quelques commentaires en caractères plus petits. Tous les ensembles considérés sont des sous-ensembles de $\mathbb{R}^N$.

Boule *Soit X_0 un point de $\mathbb{R}^N$ et r un réel positif. On appelle boule (ouverte) de centre X_0 et de rayon r, et on note $B(X_0, r)$ l'ensemble des points situés à une distance inférieure strictement à r de X_0 :*

$$B(X_0, r) = \{X \in \mathbb{R}^N : |X - X_0| < r\}$$

la distance dont il est question ici étant la distance euclidienne, notée indifféremment $|.|$ ou $\|.\|_2$.

Ouvert *Un sous-ensemble Ω de $\mathbb{R}^N$ est ouvert si, pour tout point $X_0 \in \Omega$, il existe un nombre $r > 0$ tel que la boule $B(X_0, r)$ centrée en X_0 et de rayon r soit incluse dans Ω.*

Cela généralise la notion d'intervalle ouvert sur $\mathbb{R}$. La différence est que dans $\mathbb{R}^N$, un ouvert peut avoir une forme très compliquée.

Fermé *Un sous-ensemble F de $\mathbb{R}^N$ est fermé si son complémentaire est ouvert.*
On utilise souvent une caractérisation des fermés qui utilise les suites : un ensemble F est fermé si pour toute suite de points X_n de F qui converge vers un point X, on a $X \in F$.

Fermeture (ou adhérence) *La fermeture d'un ensemble A est le plus petit fermé qui contient A. On la note $\overline{A}$.*
La fermeture est aussi l'ensemble de toutes les limites de suites d'éléments de A.

Compact *Un sous-ensemble K de $\mathbb{R}^N$ est compact si il est à la fois fermé et borné (c'est-à-dire inclus dans une boule).*
Les compacts sont très précieux en analyse :
- de toute suite de points d'un compact, on peut extraire une sous-suite convergente
- Une fonction continue (voir ci-dessous) sur un compact K et à valeurs dans $\mathbb{R}$ atteint son maximum et son minimum sur K

Fonction continue *Une fonction f de $\mathbb{R}^N$ dans $\mathbb{R}$ est continue si l'image réciproque par f de tout ouvert de $\mathbb{R}$ est un ouvert de $\mathbb{R}^N$.*
Il est plus traditionnel de définir la continuité en utilisant les limites et la notion de continuité en un point : f est continue en X_0 si $\lim_{X \to X_0} f(X) = f(X_0)$ et f est continue sur $\Omega \subset \mathbb{R}^N$, si elle est continue en tout point de Ω.

9.9.2 Calcul différentiel

9.9.2.1 Différentiabilité

Soit F une fonction définie sur un ouvert $\Omega \subset \mathbb{R}^n$ et à valeurs dans $\mathbb{R}^p$. On rappelle qu'on dit que F est différentiable en un point $X_0 \in \Omega$ si F admet un développement limité à l'ordre 1 au voisinage de X_0, c'est-à-dire si il existe une application linéaire $dF_{X_0} : \mathbb{R}^n \to \mathbb{R}^p$ telle que

$$\forall H \in \mathbb{R}^n,\ F(X_0 + H) = F(X_0) + dF_{X_0}(H) + \|H\|\varepsilon(H) \qquad \text{avec } \varepsilon(H) \to 0 \text{ quand } H \to 0.$$

Cette notion de différentiabilité se généralise exactement de la même façon pour une fonction F définie d'un espace vectoriel normé E_1 vers un espace vectoriel normé E_2 (même si ceux-ci sont de dimension infinie).

9.9.2.2 Dérivées partielles

Reprenons le cas d'une fonction F définie sur un ouvert $\Omega \subset \mathbb{R}^n$ et à valeurs dans $\mathbb{R}^p$. On notera $F(X) = (f_1(X), f_2(X), \ldots, f_p(X))$ les compsantes du vecteur image. La **dérivée partielle** de la fonction f_i par rapport à la variable x_j n'est autre que la dérivée classique de f_i par rapport à x_j quand on suppose toutes les autres variables $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$ figées (ou fixes). On la note $\frac{\partial f_i}{\partial x_j}$. On a vu ci-dessus que la différentielle de F au point X_0 est une application linéaire $dF_{X_0} : \mathbb{R}^n \to \mathbb{R}^p$. Et bien cette application linéaire a pour matrice, dans les bases canoniques de $\mathbb{R}^n$ et $\mathbb{R}^p$ la matrice jacobienne $J_F(X_0)$ de toutes les dérivées partielles de F :

$$J_F(X_0) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(X_0) & \cdots & \frac{\partial f_1}{\partial x_j}(X_0) & \cdots & \frac{\partial f_1}{\partial x_n}(X_0) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_i}{\partial x_1}(X_0) & \cdots & \frac{\partial f_i}{\partial x_j}(X_0) & \cdots & \frac{\partial f_i}{\partial x_n}(X_0) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial f_p}{\partial x_1}(X_0) & \cdots & \frac{\partial f_p}{\partial x_j}(X_0) & \cdots & \frac{\partial f_p}{\partial x_n}(X_0) \end{pmatrix}$$

Notons que pour une fonction $f : \mathbb{R}^n \to \mathbb{R}$ le vecteur $\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_j}, \ldots, \frac{\partial f}{\partial x_n}$ s'appelle **gradient** de f et se note ∇f.

De la même façon qu'on a défini les dérivées partielles premières, on peut définir les dérivées partielles secondes, par exemple $\frac{\partial^2 f}{\partial x_i \partial x_j}$ signifie qu'on dérive d'abord par rapport à la variable x_i puis on dérive la fonction $\frac{\partial f}{\partial x_i}$ par rapport à x_j. Pour une fonction de classe C^2 (c'est-à-dire une fonction dont toutes les dérivées partielles d'ordre deux sont continues), on a égalité des dérivées partielles croisées :

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}.$$

La matrice des dérivées secondes d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$ a pour terme général (sur la i-ème ligne et la j-ième colonne) $\frac{\partial^2 f}{\partial x_i \partial x_j}$ et se nomme **matrice hessienne** de f. Si f est de classe C^2 la matrice hessienne est donc une matrice symétrique.

9.9.3 Convexité

La définition de la convexité, ainsi qu'une première caractérisation à l'aide de la dérivée première est donnée à la Section 9.1.3. Citons une deuxième caractérisation fort utile pour les fonctions de classe C^2 :
Soit f définie sur $\mathbb{R}^n$ et à valeurs dans $\mathbb{R}$, alors f est convexe sur l'ouvert convexe Ω si et seulement si la matrice Hessienne $H(X)$ est positive en tout point X de Ω, c'est-à-dire si $\forall X \in \Omega$, $\forall U \in \mathbb{R}^n$, $U^t H(X) U \geq 0$. Pour vérifier la positivité de la matrice hessienne, on calcule souvent ses valeurs propres. On a alors :
f est convexe si et seulement si toutes les valeurs propres de la matrice (symétrique) hessienne $H(X)$ sont positives.